

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 geográfica (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.

Seleccione una:

-  El nuevo algoritmo siempre sea más rápido
- Esta estrategia no asegura que se obtenga el camino más corto
-  El nuevo algoritmo solo será más rápido para algunas instancias del problema

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

- sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento
- puede haber nodos que no son prometedores
- las otras dos opciones son ciertas

Se desea obtener todas las permutaciones de una lista compuesta por n elementos, ¿Qué esquema es el más adecuado?

- Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante
- Ramificación y poda, puesto que con buenas funciones de cota es más eficiente para este problema que vuelta atrás
- Vuelta atrás, para este problema no hay un esquema más eficiente

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

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

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

- provocamos que las cotas optimistas pierdan eficacia
- cambiamos la función que damos a la cota pesimista
- sería necesario comprobar si las soluciones son factibles o no puesto que ramificación y poda solo genera nodos factibles

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

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

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. Como también se conocen las coordenadas geográficas de cada ciudad se quiere usar la distancia geográfica (en línea recta) entre cada par de ciudades para como cota para limitar la búsqueda en un algoritmo de vuelta atrás.

¿Qué tipo de cota sería?

 Una cota pesimista

 Una cota óptimista

No se trataría de ninguna poda puesto que es posible que esa heurística no encuentre una solución factible

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

- las otras dos opciones son verdaderas
 - 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
 - un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones

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

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

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 geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino según su distancia geográfica.

Este algoritmo voraz, ¿serviría como cota pesimista?

Seleccione una:

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

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

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

Cuando se resuelve 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 en el caso peor es la mejor que nos podemos encontrar?

- O(2^n)
- O(n^2)
- O($n!$)

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

- Puede ser exponencial con el número de alternativas por cada decisión
- Es exponencial con el número de decisiones a tomar
- Puede ser polinómica con el número de decisiones a tomar

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

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

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. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

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

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

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

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

- suele ser polinómica con el número de alternativas por cada decisión
- es siempre exponencial con el número de decisiones a tomar
- puede ser polinómica con el número de decisiones a tomar

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

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

¿Cuál es la definición correcta de $O(g)$?

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

 a

 b

c

En los algoritmos de ramificación y poda, ¿el valor de una cota pesimista es menor que el valor de una cota optimista?

- X Sí, siempre es así
- ✓ En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir
- En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir

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

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

En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

- Las dos anteriores son verdaderas
 - garantizan que no se va a explorar nunca todo el espacio de soluciones posibles
 - pueden eliminar soluciones parciales que son factibles

El uso de funciones de cota en ramificación y poda...

- garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema
- puede reducir el número de instancias del problema que pertenecen al caso peor
- transforma en polinómicas complejidades que antes eran exponenciales

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

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

En la estrategia ramificación y poda...

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

En los algoritmos de ramificación y poda...

- Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima
- Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima
- El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida

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

- El valor de una mochila
- El valor óptimo de la mochila continua correspondiente
- El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico del objeto

En los algoritmos de ramificación y poda, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

- X No, nunca es así
- En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir
- ✓ En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir

Dado un problema de optimización cualquiera, ¿la estrategia de vuelta atrás garantiza la solución óptima?

- Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado
- 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
- Sí, puesto que ese método analiza todas las posibilidades

El problema de la asignación de turnos tiene solución...

EL PROBLEMA DE LA ASIGNACION DE TURNOS.

Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asignen una prioridad. El número de alumnos es N . Se dispone una matriz cuadrada M con N filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre -1 y $N-1$) indicando dicha prioridad (un valor -1 indica que no quiere o no puede estar con la persona de la columna correspondiente, 0 indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona) . Ningún alumno puede formar grupo consigo mismo.

Se pretende encontrar una solución para satisfacer el número máximo de alumnos según su orden de preferencia. Suponiendo que la matriz M ya existe, diseñar un algoritmo que resuelva el problema de forma óptima

- Optima mediante bactracking
- Aproximada (sub-óptima) mediante voraz
- Ambas.

El tiempo de ejecución de un algoritmo de ramificación y poda depende de:

- La instancia del problema
- La función de selección de nodos para su expansión
- De ambos

Es un problema de optimización, si el dominio de las decisiones es un conjunto infinito

 Podemos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable

 Una estrategia voraz puede ser la única alternativa

Es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo soluciones

Backtracking es aplicable a problemas de selección y optimización en los que:

- El espacio de soluciones es un conjunto finito.
- En cualquiera de los casos
- El espacio de soluciones es un conjunto infinito.

Bactracking es una técnicas de resolución general de problemas basada en:

- La búsqueda sistemática de soluciones.
- La construcción directa de la solución.
- Ninguna de las anteriores

Voraz siempre da solución óptima:

- A los dos
- Al problema de la mochila sin fraccionamiento.
- Al problema de la mochila con fraccionamiento.

El problema de la mochila, ¿encuentra su solución óptima empleando la estrategia voraz?:

- ⚡ Sólo para el caso de la mochila sin fraccionamiento
- En cualquiera de los casos anteriores
- ⚡ Sólo para el caso de la mochila con fraccionamiento

El problema del viajante de comercio puede resolverse correctamente empleando estos esquemas de programación:

- Sólo programación dinámica
- Empleando cualquiera de estos: Voraz y backtracking.
- Solo backtracking

En el método Voraz, aunque las decisiones son irreversibles, podemos asegurar que:

- Siempre obtendremos una solución factible.
- Siempre obtendremos la solución óptima.
- Sólo obtendremos la solución óptima para algunos problemas.

Al aplicar backtracking obtenemos la solución óptima a un problema:

- ⚡ Siempre
- ✓ Sólo cuando el problema cumple el principio de Optimalidad.
- En algunos casos

Dado un problema resuelto mediante backtracking y mediante ramificación y poda, el coste computacional de la solución por ramificación y poda, en comparación con la de backtracking es:

- Menor
- Igual
- Mayor

El problema de la asignación de turnos tiene solución óptima empleando:

EL PROBLEMA DE LA ASIGNACION DE TURNOS.

Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asignen una prioridad. El número de alumnos es N . Se dispone una matriz cuadrada M con N filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre -1 y $N-1$) indicando dicha prioridad (un valor -1 indica que no quiere o no puede estar con la persona de la columna correspondiente, 0 indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona) . Ningún alumno puede formar grupo consigo mismo.

Se pretende encontrar una solución para satisfacer el número máximo de alumnos según su orden de preferencia. Suponiendo que la matriz M ya existe, diseñar un algoritmo que resuelva el problema de forma óptima

- Bactracking
- Voraz
- Ambos

Al aplicar vuelta atrás a la solución de problemas, obtenemos algoritmos con costes computacionales:

- Exponenciales.
- Polinómicos.
- Los dos son correctos. Depende del problema

Vuelta atrás se emplea en la resolución de problemas de optimización en los que se pretende encontrar:

- ⓠ La dos respuestas anteriores son correctas.
- ⓡ Una solución que satisfaga unas restricciones y optimice una cierta función objetivo.
- ⓢ Todas las soluciones que satisfagan unas restricciones.

El problema de la asignación de turnos tiene solución óptima voraz aplicando la siguiente estrategia:

EL PROBLEMA DE LA ASIGNACION DE TURNOS.

Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asignen una prioridad. El número de alumnos es N . Se dispone una matriz cuadrada M con N filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre -1 y $N-1$) indicando dicha prioridad (un valor -1 indica que no quiere o no puede estar con la persona de la columna correspondiente, 0 indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona) . Ningún alumno puede formar grupo consigo mismo.

Se pretende encontrar una solución para satisfacer el número máximo de alumnos según su orden de preferencia. Suponiendo que la matriz M ya existe, diseñar un algoritmo que resuelva el problema de forma óptima

- El problema no tiene solución óptima voraz
- Seleccionamos los alumnos en orden descendente de preferencia respetando las restricciones de cabida de cada turno.
- Seleccionamos los alumnos en orden ascendente de preferencia respetando las restricciones de cabida de cada turno

El problema de la mochila, ¿puede solucionarse empleando vuelta atrás?:

- Sólo para el caso de la mochila con fraccionamiento
- Sólo para el caso de la mochila sin fraccionamiento
- Se puede aplicar para ambos casos.

Dado un problema de optimización y un algoritmo Voraz que lo soluciona, ¿cuándo podemos estar seguros de que la solución obtenida será óptima?:

- Voraz siempre encuentra solución óptima.
- En ambos casos. Las dos son correctas
- Cuando demostremos formalmente que el criterio conduce a una solución óptima para cualquier instancia del problema.

El problema de la asignación de turnos resuelto mediante backtracking tiene una complejidad:

EL PROBLEMA DE LA ASIGNACION DE TURNOS.

Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asignen una prioridad. El número de alumnos es N . Se dispone una matriz cuadrada M con N filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre -1 y $N-1$) indicando dicha prioridad (un valor -1 indica que no quiere o no puede estar con la persona de la columna correspondiente, 0 indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona). Ningún alumno puede formar grupo consigo mismo.

Se pretende encontrar una solución para satisfacer el número maximo de alumnos según su orden de preferencia. Suponiendo que la matriz M ya existe, diseñar un algoritmo que resuelva el problema de forma óptima

- Exponencial
- Polinómica
- Ninguna de la dos

Bactracking procede a obtener la solución a un problema de optimización empleando la siguiente estrategia:

 Genera todas las combinaciones de la solución y selecciona la que optimiza la función objetivo.

 Genera todas las soluciones factibles y selecciona la que optimiza la función objetivo.

Genera una solución factible empleando un criterio óptimo

Dado un grafo G que representa las poblaciones de la provincia de Alicante de más de 20.000 habitantes junto con todas las carreteras de conexión entre ellas. Queremos obtener el recorrido que nos permita pasar por todas estas ciudades una única vez y volver al punto de origen recorriendo el mínimo número de kilómetros. Si aplicamos una estrategia voraz sobre este grafo obtendremos...

- Puede que no encuentre ninguna solución aunque ésta exista
- Una solución factible
- La solución óptima

Si aplicamos un algoritmo voraz que no nos garantiza la solución óptima sobre un problema entonces...

- ⚡ Si el problema tiene solución óptima, el esquema voraz nos garantiza que la encuentra
- Obtenremos una solución factible.
- Puede que no encuentre ninguna solución aunque ésta exista.

El método voraz se emplea en la resolución de problemas de selección y optimización en los que se pretende encontrar:

- La dos respuestas anteriores son correctas.
- Todas las soluciones que satisfagan unas restricciones.
- Una solución que satisfaga unas restricciones y optimice una cierta función objetivo.

Si aplicamos un esquema backtracking que no nos garantiza la solución óptima sobre un problema entonces

-  Obtenremos una solución factible.
- Puede que no encuentre ninguna solución aunque ésta exista.
-  Ninguna de las anteriores.

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

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

Dada la solución recursiva mediante vuelta atrás al problema de la asignación de turnos ¿cuántas nuevas llamadas recursivas genera cada llamada recursiva?

EL PROBLEMA DE LA ASIGNACION DE TURNOS.

Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asignen una prioridad. El número de alumnos es N . Se dispone una matriz cuadrada M con N filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre -1 y $N-1$) indicando dicha prioridad (un valor -1 indica que no quiere o no puede estar con la persona de la columna correspondiente, 0 indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona) . Ningún alumno puede formar grupo consigo mismo.

Se pretende encontrar una solución para satisfacer el número máximo de alumnos según su orden de preferencia. Suponiendo que la matriz M ya existe, diseñar un algoritmo que resuelva el problema de forma óptima

- una o dos
- una o ninguna
- ✓ ninguna de las anteriores

¿Cuál de estas afirmaciones es falsa?

- La complejidad en el peor caso de las soluciones Backtracking y ramificación y poda a un mismo problema es la misma.
- Para un mismo problema, ramificación y poda explora siempre un número menor o igual que backtracking
- Backtracking inspecciona todo el espacio de soluciones de un problema mientras que Ramificación y poda no.

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?

- Cambiamos la función que damos a la cota pesimista
- La comprobación de las soluciones factibles: en ramificación y poda no es necesario puesto que sólo genera nodos factibles
- Aprovechamos mejor las cotas optimistas

Bactracking genera las soluciones posibles al problema:

- ⓠ Mediante el recorrido en profundidad del árbol que representa el espacio de soluciones.
- ⓡ Mediante el recorrido en anchura del árbol que representa el espacio de soluciones
- ⓢ Ninguna de las anteriores

En un problema resuelto por backtracking, el conjunto de valores que pueden tomar las componentes de la tupla solución, ha de ser:

- Infinito.
- continuo
- finito

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

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

El uso de funciones de cota en ramificación y poda...



- puede reducir el número de instancias del problema que pertenecen al caso peor.
- garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.
- transforma en polinómicas complejidades que antes eran exponenciales.

Sí un problema de optimización lo es para una función que toma valores continuos ...

- La programación dinámica iterativa siempre es mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.
- 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.
- La programación dinámica recursiva puede resultar mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.

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

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

Sea la siguiente relación de recurrencia

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

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

✗

$g(n) = 1$

✓

$g(n) = n^2$

$g(n) = n$

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.

-  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.
-  Se puede mejorar la eficiencia del algoritmo convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.
-  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.

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?

- $O(n^3)$
- $O(2^n)$
- $O(n^2)$

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?

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

Uno de estos tres problemas no tiene una solución eficiente que siga el esquema de programación dinámica

- El Problema de la mochila discreta.
- El Problema de las torres de Hanoi
- 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.

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz (greedy) que es óptima?

- El problema de la mochila discreta.
- El árbol de cobertura de coste mínimo de un grafo conexo.
- El problema de la mochila continua o con fraccionamiento.

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).

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

Dado un problema de optimización cualquiera, ¿la estrategia de vuelta atrás garantiza la solución óptima?

- 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.
- Sí, puesto que ese método analiza todas las posibilidades.
- Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.

¿Para cuál de estos problemas de optimización existe una solución voraz?

- El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
- El problema de la mochila discreta.
- 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.

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

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

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

- Sí, pero siempre que la complejidad temporal conjunta de las operaciones de descomposición del problema y la combinación de las soluciones sea polinómica.
- No
- Sí, en cualquier caso.

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

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

Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

- Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
- 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.
- 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.

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.

- El nuevo algoritmo no garantiza que vaya a ser más rápido para todas las instancias del problema posibles.
- Esta estrategia no asegura que se obtenga el camino mas corto.
- El nuevo algoritmo siempre sera más rápido.

En los algoritmos de ramificación y poda

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

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

....

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

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

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

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...

- Las otras dos opciones son ciertas.
- puede ser relevante si se utilizan mecanismos de poda basados en estimaciones optimistas.
- es irrelevante si no se utilizan mecanismos de poda basados en la mejor solución hasta el momento.

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

- ...divide y vencerás.
- ...voraz.
- ...ramificación y poda.

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.

- 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.
 - La complejidad temporal de la mejor solución posible al problema es $O(n^2)$.
 - La complejidad temporal de la mejor solución posible al problema es $O(n!)$.

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?

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

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

- ⚗ siempre obtiene la solución óptima.
- ⚗ siempre obtiene una solución factible.
- ⚗ garantiza la solución óptima sólo para determinados problemas.

Di cuál de estos tres algoritmos no es un algoritmo de "divide y vencerás"

 Quicksort

Mergesort

 El algoritmo de Prim

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?

- No, ya que en cualquier caso se deben explorar todas las soluciones factibles.
- Sí, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.
- Sí, tanto si se usan cotas optimistas para podar el árbol de búsqueda como si no.

Dadas las siguientes funciones: (imagen)

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

```
// 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() );
}
```



cuadrática

exponencial

cúbica

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

- es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione.
- una estrategia voraz puede ser la única alternativa.
- podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.

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?

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

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

- La mochila discreta sin restricciones adicionales.
- El problema del cambio.
- El problema de la asignación de tareas.

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

- p,r,n-r[n]
- p,r-1,n
- p,r,n-i

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

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)

- X No, nunca es así.
- En general si, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
- ✓ En general si, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.

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 $f(n) \in O(n^2)$

 a

 b

c

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que...

- en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.

El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.

- en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos.

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

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

En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

- Las otras dos opciones son ciertas.
- garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- pueden eliminar soluciones parciales que son factibles.

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

- ...nunca tendrá una complejidad exponencial.
- Las dos anteriores son ciertas.
- ...será más eficiente cuanto más equitativa sea la división en subproblemas.

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?

- Sí, ya que el mergesort es $O(n \log n)$ y este es $O(n)$
- No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.
- Sólo si $d \log d > kn$ (donde k es una constante que depende de la implementación)

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

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

En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

- Pueden eliminar vectores que representan posibles soluciones factibles.
- Garantizan que no se va a explorar todo el espacio de soluciones posibles.
- Las otras dos opciones son ambas verdaderas...

Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?

- Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
- Vuelta atrás, es el esquema más eficiente para este problema.
- Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.

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

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

El esquema voraz...

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

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

- Las otras dos opciones son ambas verdaderas.
- ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a .
- ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.

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

- Programación dinámica
- Divide y vencerás.
- Ramificación y poda.

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

- No, para garantizar que se encuentra un camino de salida hay que aplicar métodos de búsqueda exhaustiva como vuelta atrás o ramificación y poda.
- Si, en caso de existir con este esquema siempre se puede encontrar un camino de salida
- 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.

Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es par y que se utilizará la siguiente recurrencia:

$\text{pot.}(x, n) = \text{pot.}(x, n/2) * \text{pot.}(x, n/2)$; ¿Qué esquema resulta ser más eficiente en cuanto al coste temporal?

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

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

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

Una de las prácticas de laboratorio consistió en el cálculo empírico de la complejidad temporal promedio del algoritmo de ordenación de vectores Quicksort tomando como centinela el elemento del vector que ocupa la posición central. ¿Cuál es el orden de complejidad que se obtuvo?

 $\Theta(n^2)$

 $O(n \log n)$

$O(n \log^2 n)$

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

- \bullet $g(n) \in O(f(n))$ pero $f(n) \notin O(g(n))$
- \bullet $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
- \bullet $f(n) \in O(g(n))$ pero $g(n) \notin O(f(n))$

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?

- $\text{nc}(n, m) = \text{nc}(n - 1, m) + \text{nc}(n, m - 1) + \text{nc}(n - 1, m - 1)$
- $\text{nc}(n, m) = \text{nc}(n - 1, m) * \text{nc}(n, m - 1) * \text{nc}(n - 1, m - 1)$
- Ninguna de las otras dos recurrencias se corresponde con un esquema de divide y vencerás.

Sea la siguiente relación de recurrencia: (imagen)

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

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

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

El esquema de vuelta atrás...

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

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

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

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?

- 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.
- De una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización.
- De una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización.

Dado el problema del laberinto con tres movimientos, se pretende conocer la longitud del camino de salida más corto. Para ella 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?

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

(imagen)

Dada la siguiente función:

```
int exa (vector<int>& v){  
    int i=0, 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[i]=x;  
        i++;  
    } while (i<n);  
    return 0;  
}
```

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

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

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

Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones de recurrencia expresa mejor su complejidad temporal para el caso general, siendo n el número de discos?

- \bullet $T(n) = T(n - 1) + n$
- \circ $T(n) = 2T(n - 1) + 1$
- \circ $T(n) = 2T(n - 1) + n$

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

- \bullet $T(n) \in O(n)$ y $T(n) \in \Theta(n)$
- \bullet $T(n) \in \Theta(n)$ y $T(n) \in \Omega(n^2)$
- \bullet $T(n) \in \Omega(n)$ y $T(n) \in \Theta(n^2)$

Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc... Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?

- ⚡ Un vector de booleanos.
- Una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.
- Un par de enteros que indiquen los cortes realizados y el valor acumulado.

¿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 n * f(n-1,m) + f(n-2,m);  
}
```

- $\Theta(n^2)$
- $\Theta(n \times m)$
- $\Theta(n)$

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.

(imagen)

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

El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continue ordenado es:

- $O(\log n)$
- $\Omega(n^2)$
- $O(n)$

Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?

- Que se podría explorar menos nodos de los necesarios.
- Que se podría podar el nodo que conduce a la solución óptima.
- Que se podría explorar más nodos de los necesarios.

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

- $f \in \Omega(g_1 * g_2)$
- $f \in \Omega(\min(g_1, g_2))$
- $f \in \Omega(g_1 + g_2)$

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

- $O(n \log n)$
- $O(n^3)$
- $O(n^3 \log n)$

En el problema del viajante de comercio (travelling salesman problem) queremos listar todas las soluciones factibles.

- Lo más importante es conseguir una cota pesimista adecuada.
- X Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.
- ✓ El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
- La 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.

(imagen)

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

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

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

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

- Cola de prioridad
- Cola
- Pila

(imagen)



O(n)

O(log n)

O(n^2)

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?

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

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

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.

$n + n \log n \in \Omega(n)$

$O(2^{(\log n)}) < O(n^2)$

$\Theta(n) < \Theta(n^2)$

¿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?

- El coste temporal promedio
- El coste temporal asintótico en el caso medio
- Nada de interés

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

$\Theta(n^2)$

$\Theta(n \log n)$

Ninguna de las otras dos opciones es cierta

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

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

De las siguientes afirmaciones marca la que es verdadera.

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

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

- ⚡ Siempre es mayor o igual que la mejor solución pasible alcanzada.
- ⚡ Asegura un ahorro en la comprobación de todas las soluciones factibles.
- ⚡ Las otras dos opciones son ambas falsas.

Dada la siguiente función (imagen)
marca la respuesta correcta:

```
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;
}
```

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

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

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

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



a

b

c

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

- $\Omega(n)$ y $O(n^2)$
- $O(n)$
- $O(n \log n)$

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

- exponencial en el caso peor
- cuadrática en el caso peor
- exponencial en cualquier caso

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...

- se van a utilizar tantos colores distintos a los ya utilizados como vértices quedan por colorear
- solo va a ser necesario un color más
- no se van a utilizar colores distintos a los ya utilizados

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

- no presenta caso mejor y peor para instancias del mismo tamaño
- se comporta peor cuando el vector ya está ordenado
- se comporta mejor cuando el vector ya está ordenado

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

- f(n) ∈ O(n^3)
- f(n) ∈ Θ(n^2)
- f(n) ∈ Θ(n^3)

Tenemos un conjunto de n enteros positivos y queremos encontrar el subconjunto de tamaño m de suma mínima...

- Una técnica voraz daría una solución óptima.
 - 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.
- 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.

¿Qué algoritmo es asintóticamente más rápido, Quicksort o Mergesort?

- Son los dos igual de rápidos, ya que el coste temporal asintótico de ambos es $O(n \log n)$
- Como su nombre indica, el Quicksort
- El Mergesort es siempre más rápido o igual (salvo una constante) que el Quicksort

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

- \bullet $f(n) \in \Theta(n^2)$
- \bullet $f(n) \in \Theta(n \log n)$
- \bullet $f(n) \in \Theta(n)$

¿Cuál de estas afirmaciones es cierta?

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

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

\bullet $g(n) \in \Omega(n^k)$

\circ Las otras dos afirmaciones son falsas.

\circ $g(n) \in \Theta(n^k)$

Indicad cuál de estas tres expresiones es falsa:

 $\Theta(n) \subset \Theta(n^2)$

 $\Theta(n) \subset O(n)$

 $\Theta(n/2) = \Theta(n)$

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

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

✓ Es $\Theta(n^2)$

Es $\Theta(n)$

Es $O(n^2)$ pero no $\Omega(n^2)$

¿Pertenece $3n^2 + 3$ a $O(n^3)$?

 No

 Sí

Sólo para $c = 1$ y $n_0 = 5$

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

- un recorrido guiado por estimaciones de las mejores ramas del árbol que representan el espacio de soluciones
- 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
- un recorrido en profundidad del árbol que representa el espacio de soluciones

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

- f(n) ∈ Θ(g(n))
- g(n) ∈ O(f(n))
- f(n) ∈ O(g(n))

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

 $f \in \Theta(g_1 * g_2)$

 $f \in \Theta(g_1 + g_2)$

$f \in \Theta(\max(g_1, g_2))$

¿Cuál es el coste espacial asintótico del siguiente algoritmo?

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

- O(n)
- O(1)
- O(log n)

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);  
}
```

- Se puede reducir hasta lineal
- Cuadrática
- La función no cumple con los requisitos necesarios para poder aplicar programación dinámica

¿Cuál de estas afirmaciones es falsa?

- 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
- Todos los problemas de optimización tienen una solución voraz que es óptima sea cual sea la instancia a resolver
- Hay problemas de optimización para los cuales se puede obtener siempre la solución óptima utilizando una estrategia voraz

Dada la relación de recurrencia:

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

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

- p > a^k
- p < a^k
- p = a^k

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

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

La complejidad temporal (o coste temporal asintótico) en el mejor de los casos...

- es una función de la talla, o tamaño del problema, que tiene que estar definida para todos los posibles valores de ésta.
- es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar
- las dos anteriores son verdaderas.

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



- Es condición necesaria (aunque no suficiente) que el dominio de las decisiones sea discreto o discretizable.
- Es condición necesaria y suficiente que el dominio de las decisiones sea discreto o discretizable.
- 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.

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

Di cuál de las siguientes afirmaciones es cierta:

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

- Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort
- Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal en el caso peor del algoritmo de ordenación Quicksort
- Si $g(n) \in \Theta(1)$ la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort

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

- ⚡ Nunca tendrá una complejidad exponencial
- ✓ Será más eficiente cuanto más equitativa sea la división en subproblemas
- Las demás opciones son verdaderas

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

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

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

- O($n \log n$)
- O(2^n)
- O(n^2)

Estudiad la relación de recurrencia:

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

(donde p y q son enteros mayores que 1). Di cuál de los siguientes esquemas algorítmicos produce de manera natural relaciones de recurrencia así.

- Programación dinámica
- Divide y vencerás
- Ramificación y poda

Cogemos el algoritmo de Mergesort y en lugar de dividirlo el vector en dos partes, lo dividimos en tres. Posteriormente combinamos las soluciones parciales. ¿Cuál sería la complejidad temporal asintótica de la combinación de las soluciones parciales?

$\Theta(\log n)$

$\Theta(n)$

Ninguna de las otras dos opciones es cierta

Uno de estos tres problemas no tiene una solución trivial y eficiente que siga el esquema voraz

- El problema de la mochila continua
- El problema de la mochila discreta sin limitación en la carga máxima de la mochila
- El problema del cambio

¿Qué se entiende por "tamaño del problema"?

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

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

- f(n) ∈ Θ(n²)
- f(n) ∈ Ω(n³)
- f(n) ∈ Θ(n³)

Las relaciones de recurrencia...

- sirven para reducir el coste temporal de una solución cuando es prohibitivo
- expresan recursivamente el coste temporal de un algoritmo
- aparecen sólo cuando la solución sea del tipo divide y vencerás

Los algoritmos de ordenación Quicksort y Mergesort tienen en común...

- ... que se ejecutan en tiempo $O(n)$.
- ... que aplican la estrategia de divide y vencerás.
- ... que ordenan el vector sin usar espacio adicional.

¿Cuál de estas tres expresiones es cierta?



$O(2^{\log n}) \subset O(n^2) \subset O(2^n)$

$O(n^2) \subset O(2^{\log n}) \subseteq O(2^n)$

$O(n^2) \subset O(2^{\log n}) \subset O(2^n)$

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

O(n log n)

O(n^2)

O(n)

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.

- X No se puede usar vuelta atrás porque las decisiones no son valores discretos.
- ✓ No hay ningún problema en usar una técnica de vuelta atrás.
- No se puede usar vuelta atrás porque el número de combinaciones es infinito.

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

```
void examen (vector <int> & v) {
    int i=0, j, 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]){
                swap(v[j], v[j-1]);
                permuta=1;
            }
    }
}
```

- Esta función no tiene caso mejor
- $\Omega(n)$
- $\Omega(1)$

¿Cuál de estas afirmaciones es falsa?

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

Sea la siguiente relación de recurrencia

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

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

✗

$g(n) = n^2$

✓

$g(n) = 1$

$g(n) = n$

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.

(imagen)

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

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

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



a

b

c

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

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

X

a

✓

b

c

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

- es siempre exponencial con el número de decisiones a tomar
- puede ser polinómica con el número de decisiones a tomar
- suele ser polinómica con el número de alternativas por cada decisión

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



Mediante un vector de booleanos

Mediante un vector de reales

Este problema no se puede resolver usando ramificación y poda si



no se fija una cota superior al número total de aplicaciones de funciones

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?

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

¿Cuál de estas expresiones es falsa?

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

Dada la relación de recurrencia:

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

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

- p > a^k
- p = a^k
- p < a^k

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

- Según el tipo de cota, las pesimistas puede que no descarten
- ning n nodo pero el uso de cotas optimistas garantiza la reducci n del espacio de b queda
 - No, las cotas tanto optimistas como pesimistas garantizan la reducci n del espacio de soluciones y por tanto la eficiencia del algoritmo
 - S , puesto que es posible que a pesar de utilizar dichas cotas no se descarte ning n nodo

Di cuál de estos resultados de coste temporal asintótico es falsa:

- La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en $O(\log n)$.
- La ordenación de un vector usando el algoritmo Quicksort requiere en el peor caso un tiempo en $\Omega(n^2)$
- La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en $\Omega(n^2)$

Un problema se puede resolver por Divide y Vencerás siempre que:

- Cumpla el principio de optimalidad
- Cumpla el teorema de reducción
- Ninguna de las anteriores

¿Con qué esquema de programación obtenemos algoritmos que calculan la distancia de edición entre dos cadenas?

- Programación Dinámica
- Divide y Vencerás
- Ambos

¿Qué esquema de programación es el adecuado para resolver el problema de la búsqueda binaria?

- ⚡ Divide y Vencerás
- Programación Dinámica
- Ninguno de los dos

¿Cuándo utilizaremos Programación Dinámica en lugar de Divide y Vencerás?

- Cuando se reduce el coste espacial
- Cuando se incrementa la eficiencia
- Cuando se incrementa la eficacia

Si n es el número de elementos del vector, el coste del algoritmo Mergesort es:

- $\Theta(n^2)$ y $\Omega(n \log(n))$
- $\Theta(n \log(n))$
- $\Theta(n^2)$

Dado el algoritmo de búsqueda binaria, supongamos que, en vez de dividir la lista de elementos en dos mitades del mismo tamaño, la dividamos en dos partes de tamaños $1/3$ y $2/3$. El coste de este algoritmo:

- ⚡ Es menor que el del original
- ✅ Es mayor que el del original
- ⚡ Es el mismo que el original

Para que un problema de optimización se pueda resolver mediante programación dinámica es necesario que:

- Cumpla el principio de optimalidad
- Cumpla el teorema de reducción
- Cumpla las dos anteriores

La serie de números de Fibonacci se define de la siguiente forma: (imagen)

¿Qué implementación de entre las siguientes supone el menor coste?

$$fib(n) = \begin{cases} 1 & n \leq 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$$

- ⚡ Divide y vencerás
- ✅ Programación dinámica
- Ambas tienen el mismo coste asintótico

En programación dinámica, dónde almacenaremos los valores de los problemas resueltos?

- En un vector bidimensional
- Depende del problema
- En un vector unidimensional

¿Qué esquema de programación es el adecuado para resolver el problema k-ésimo mínimo en un vector?

- X Programación Dinámica
- ✓ Divide y Vencerás
- Ninguno de los dos

¿Qué esquema algorítmico utiliza el algoritmo de ordenación Quicksort?

- Programación Dinámica
- Divide y Vencerás
- Backtracking

Dada la solución recursiva al problema de encontrar el k -ésimo mínimo de un vector. Cada llamada recursiva, ¿cuántas nuevas llamadas recursivas genera?

- ⚗ dos o ninguna
- ⚗ una o dos
- ⚗ una o ninguna

Si aplicamos programación dinámica a un problema que también tiene solución por divide y vencerás podemos asegurar que...

- El coste temporal se reduce y el espacial aumenta con respecto a la solución por DyV
- El coste temporal aumenta y el espacial se reduce con respecto a la solución por DyV
- Ninguna de las anteriores

Ante un problema que presenta una solución recursiva siempre podemos aplicar:

- Divide y vencerás
- Programación Dinámica
- Cualquiera de las dos anteriores

La serie de números de Fibonacci se define de la siguiente forma: (imagen)

Para implementar esta función podemos emplear...

$$fib(n) = \begin{cases} 1 & n \leq 1 \\ fib(n - 1) + fib(n - 2) & n > 1 \end{cases}$$

- X ○ Divide y vencerás
- Programación dinámica
- ✓ ○ Cualquiera de las dos anteriores

La solución al problema de encontrar el k -ésimo mínimo de un vector pone en práctica la siguiente estrategia:

- No ordena ningún elemento del vector
- Ordena parcialmente el vector
- Ordenar totalmente el vector

En cual de los siguientes casos no se puede aplicar el esquema Divide y Vencerás:

- Cuando los subproblemas son de tamaños muy diferentes
- Cuando el problema no cumple el principio de optimalidad
- Se puede aplicar en ambos casos

La programación dinámica, para resolver un problema, aplica la estrategia...

- Se resuelven los problemas más pequeños y, combinando las soluciones,
se obtienen las soluciones de problemas sucesivamente más grandes
hasta llegar al problema original
- Se descompone el problema a resolver en subproblemas más pequeños,
que se resuelven independientemente para finalmente combinar las
soluciones de los subproblemas para obtener la solución del problema
original
- Ninguna de las anteriores

Supongamos el problema de la mochila resuelto mediante Programación Dinámica y particularizado para n elementos y un peso máximo trasportable de P . ¿Es necesario calcular valores para toda la matriz auxiliar para obtener el resultado?

 Si

 No

Depende de los valores de n y P

Disponemos de dos cadenas de longitudes m y n . Si resolvemos el problema de la distancia de edición mediante programación dinámica, ¿De qué tamaño debemos definir la matriz que necesitaremos?

(m-1) x (n-1)

m x n

(m+1) x (n+1)

Si n es el número de elementos de un vector. La solución de menor coste al problema de encontrar su k -ésimo mínimo tiene la siguiente complejidad:

- $\Omega(n)$ y $O(n \log(n))$
- $\Omega(n)$ y $O(n^2)$
- Ninguna de las dos

Si n es el número de elementos de un vector. Podemos encontrar una solución al problema de encontrar su k -ésimo que esté acotada superiormente por:

- O(n^3)
- O(n)
- Ninguna de las anteriores

Dada una solución recursiva a un problema, ¿Cómo podemos evitar la resolución de los mismos subproblemas muchas veces?

- Resolver los subproblemas de mayor a menor y guardar su resultado en una tabla, inicializándola con los problemas pequeños
- Resolver los subproblemas de mayor a menor y guardar su resultado en una tabla, inicializándola con los problemas más grandes
- Resolver los subproblemas de menor a mayor y guardar su resultado en una tabla, inicializándola con los problemas pequeños

Un problema de optimización cuya solución se puede expresar mediante una secuencia de decisiones cumple el principio de optimalidad si, dada una secuencia óptima:

- Existe al menos una subsecuencia de esa solución que corresponde a la solución óptima de su subproblema asociado
- Existe una subsecuencia de esa solución que corresponde a la solución óptima de su subproblema asociado
- Cualquier subsecuencia de esa solución corresponde a la solución óptima de su subproblema asociado

Si n es el número de elementos de un vector. La solución de menor coste al problema de la búsqueda binaria tiene la siguiente complejidad:

- $\Omega(\log(n))$ y $O(n \log(n))$
- $\Omega(1)$ y $O(\log(n))$
- $\Theta(n \log(n))$

¿Qué esquema de programación es el adecuado para resolver el problema k-ésimo mínimo en un vector?

- Programación Dinámica
- Divide y Vencerás
- Ninguno de los dos

El problema de la mochila, ¿Puede solucionarse de forma óptima empleando la estrategia de divide y vencerás?

- Sólo para el caso de la mochila con fraccionamiento
- Sólo para el caso de la mochila sin fraccionamiento
- Sí, se puede aplicar para ambos casos

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

- Ninguna de las otras dos opciones es correcta
- Constante con el tamaño del vector
- $O(n)$

Di cuál de estos resultados de coste temporal asintótico es falsa:

- La ordenación de un vector usando el algoritmo Quicksort requiere en el peor caso un tiempo en $\Omega(n^2)$
- La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en $\Omega(n^2)$
- La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en $O(\log(n))$

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

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

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

$\Theta(n^2) \subset \Theta(n^3)$

$\Omega(n^2) \subset \Omega(n^3)$

$O(n^2) \subset O(n^3)$

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

- \bullet $f(n) \in \Theta(n)$
- \circ $f(n) \in \Theta(n \log(n))$
- \circ $f(n) \in \Theta(n^2)$

Se dispone de un vector de tamaño n cuyos elementos están de antemano organizados formando un montículo (heap). ¿Cuál sería la complejidad temporal de reorganizar el vector para que quede ordenado?

- ⚡ $O(\log(n))$
- ⚡ $O(n)$
- ⚡ $O(n \log(n))$

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

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

- O($n \log(n)$)
- O(n)
- O(n^2)

Sea $f(n)$ la solución de la relación de recurrencia... (imagen)

Indicad cuál de estas tres expresiones es cierta:

$$f(n) = 2f(n - 1) + 1; f(1) = 1.$$

- \bullet $f(n) \in \Theta(n^2)$
- \circ $f(n) \in \Theta(2^n)$
- \circ $f(n) \in \Theta(n)$

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

- $\Theta(n^2 \log(n))$
- $\Theta(n^2)$
- $\Theta(2^n)$

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

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

- $O(n^2)$
- $O(n\sqrt{n})$
- $O(n \log(n))$

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

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

✗ $\Theta(2^n)$

$\Theta(n^3 \log(n))$

✓ $\Theta(n^3)$

Dado el siguiente fragmento de código... (imagen)

Indica cuál de las siguientes expresiones es la que mejor refleja su complejidad temporal.

```
for( i=0; i<n; i++ ) for( j=1; j<i; j+=2 ) s+=i*j;
```

- $\sum \log(j)$
- Ninguna de las otras dos opciones son correctas
- $\sum \log(i)$

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

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

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

- \bullet $f(n) \in \Theta(n)$
- $f(n) \in \Theta(n \log(n))$
- $f(n) \in \Theta(\log(n))$

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

```
k=n/4;  
for( int i = k; i < n - k; i++) {  
    A[i] = 0;  
    for( int j = i - k; j < i + k; j++ )  
        A[i] += B[j];  
}
```

- O(n^2)
- O(n)
- O($n \log(n)$)

Un problema de tamaño n puede transformarse en tiempo $O(n^3)$ en ocho de tamaño $n/2$; 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?

- O(n^3)
- O($n^2 \log(n)$)
- O($n^3 \log(n)$)

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

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

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

- $O(n^2) \subset O(n^3)$
- $\Omega(n^3) \subset \Omega(n^2)$
- $\Theta(n^2) \subset \Theta(n^3)$

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

$\Omega(n^2) \subset \Omega(n^3)$

$\Theta(n^2) \subset \Theta(n^3)$

$O(n^2) \subset O(n^3)$

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

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

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

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

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

- $\Theta(\log^2(n)) = \Theta(\log(n^2))$
- $\Theta(\log^2(n)) = \Theta(\log(n))$
- $\Theta(\log(n)) = \Theta(\log(n^2))$

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

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

X $f(n) \in \Theta(n^2 \log(n))$

✓ $f(n) \in \Theta(n^2)$

$f(n) \in \Theta(n)$

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

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

- f(n) ∈ Θ(n log(n))
- f(n) ∈ Θ(n^3)
- f(n) ∈ Θ(n)

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

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

 $O(n)$

 $O(n \log(n))$

$O(n^2)$

Se pretende implementar mediante programación dinámica iterativa la función recursiva: (imagen)

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

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

- O(y^2)
- O(y)
- O(1)

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que...

- ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.
 - ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos.
- El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que...

- el número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica
- en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos
- en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos

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

- Un algoritmo de programación dinámica
- Un algoritmo de estilo Divide y Vencerás
- Un algoritmo voraz

La programación dinámica...



- Las otras dos opciones son ciertas

- En algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos
 - Normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores

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

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

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

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

Se pretende implementar mediante programación dinámica iterativa la función recursiva: (imagen)

¿Cuál es la mejor estructura para el almacén?

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

int A

int A [] []

int A[]

Se pretende implementar mediante programación dinámica iterativa la función recursiva: (imagen)

¿Cuál es la mejor estructura para el almacén?

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

int A

int A[] []

int A[]

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

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

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

- El árbol de cobertura de coste mínimo de un grafo conexo
- El problema de la mochila continua o con fraccionamiento
- El problema de la mochila discreta o sin fraccionamiento

Se pretende implementar mediante programación dinámica iterativa la función recursiva: (imagen)

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

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

 O(x)

 O(1)

O(x^2)

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

- X no se puede resolver en general con una estrategia voraz
- sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo
- se puede resolver siempre con una estrategia voraz

En el método voraz...

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

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

- Meter primero los elementos de mayor valor
- Meter primero los elementos de menor peso
- Meter primero los elementos de mayor valor específico o valor por unidad de peso

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

- al menos una solución factible está garantizada
- la solución óptima está garantizada
- ninguna de las otras dos opciones es cierta

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

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

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

Di cuál de estas tres afirmaciones es FALSA.

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

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



Ya no se garantizaría la solución óptima pero sí una factible

Habría que replantearse el criterio de selección para comenzar por

aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible



La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna

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

- El método voraz
- Programación dinámica
- Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño

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

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

Supongamos que un informático quiere subir a una montaña y como no es una persona normal 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?

- Un algoritmo divide y vencerás
- Un algoritmo de programación dinámica
- Un algoritmo voraz

Los algoritmos de programación dinámica hacen uso...

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

La eficiencia de los algoritmos voraces se basa en el hecho de que...

- antes de tomar una decisión se comprueba si satisface las restricciones del problema
- las decisiones tomadas nunca se reconsideran
- con antelación, las posibles decisiones se ordenan de mejor a peor

Se pretende implementar mediante programación dinámica iterativa la función recursiva: (imagen)

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

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

- O(y^2)
- O(y)
- O(1)

Dada la suma de la recurrencia: (imagen)

¿cuál de las siguientes afirmaciones es cierta?

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

- T(n) ∈ Θ(2^n)
- T(n) ∈ Θ(n!)
- T(n) ∈ Θ(n^2)

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

- ⚡ siempre obtiene una solución factible
- ⚡ siempre obtiene la solución óptima
- ⚡ garantiza la solución óptima sólo para determinados problemas

¿Cuál de estas estrategias para calcular el n -ésimo elemento de la serie de Fibonacci (imagen) es más eficiente?

$$(f(n) = f(n-1) + f(n-2), f(1) = f(2) = 1)$$

- X La estrategia voraz
- Las dos estrategias citadas serían similares en cuanto a eficiencia
- ✓ Programación Dinámica

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

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

Se pretende implementar mediante programación dinámica iterativa la función recursiva: (imagen)

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

```
unsigned f( unsigned x, unsigned v[] ) {  
    if (x==0)  
        return 0;  
  
    unsigned m = 0;  
  
    for ( unsigned k = 1; k < x; k++ )  
        m = max( m, v[k] + f( x-k, v ) );  
  
    return m;  
}
```

- O(x^2)
- O(x)
- O(1)

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

- ninguna de las otras dos opciones es cierta
- meter primero los elementos de mayor valor específico o valor por unidad de peso
- meter primero los elementos de mayor valor

Los algoritmos de programación dinámica hacen uso ...

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

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

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

Se pretende implementar mediante programación dinámica iterativa la función recursiva: (imagen)

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

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

- O(x^2)
- O(y)
- O(y^2)

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

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

En el método voraz...

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

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que...

- En la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos
- En la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos
- El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica

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

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

¿Por qué se utiliza el TAD "Union-find" en el algoritmo de Kruskal?

- Para comprobar si dos vértices son equivalentes
- Para comprobar si un arco forma ciclos
- Para comprobar si un vértice ya ha sido visitado

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

- El árbol de cobertura de coste mínimo de un grafo conexo
- El problema de la mochila discreta o sin fraccionamiento
- El problema de la mochila continua o con fraccionamiento

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

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

Se pretende implementar mediante programación dinámica iterativa la función recursiva: (Imagen)

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

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

- O(x)
- O(x^*y)
- O(v)

La solución óptima al problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado...

- Debe construirlo vértice a vértice: arista a arista no puede ser
- Puede construirlo tanto vértice a vértice como arista a arista
- Debe construirlo arista a arista: vértice a vértice no puede ser

¿Qué mecanismo se usa para acelerar el algoritmo de Prim?

- Mantener para cada vértice su "padre" más cercano
- Mantener una lista de los arcos ordenador según su peso
- El TAD "Union-find"

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

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

La programación dinámica...

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

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

- La estrategia voraz
- Programación dinámica
- Las dos estrategias citadas serían similares en cuanto a eficiencia

Se pretende aplicar la técnica memoización a la siguiente función recursiva: (imagen)

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

```
int f( int x, int y ) {  
    if( x > y ) return 1;  
    return x*(y-1) + f(x,y-2);  
}
```

- ⚪ Temporal $O(x*y)$ y espacial $O(x)$
- ⚫ Ninguna de las otras dos opciones es correcta
- ⚫ $O(y - x)$, tanto temporal como espacial

La eficiencia de los algoritmos voraces se basa en el hecho de que...

- antes de tomar una decisión se comprueba si satisface las restricciones del problema
- con antelación, las posibles decisiones se ordenan de mejor a peor
- las decisiones tomadas nunca se reconsideran

Se pretende implementar mediante programación dinámica iterativa la función recursiva: (imagen)

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

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

- O(x^*y)
- O(y)
- O(x)

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

- El fontanero diligente y el problema del cambio
- El fontanero diligente y la mochila continua
- El fontanero diligente y la asignación de tareas

Se pretende implementar mediante programación dinámica iterativa la función recursiva: (imagen)

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

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

- O(y^2)
- O(x^2)
- O(y)

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

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

Se pretende implementar mediante programación dinámica iterativa la función recursiva: (imagen)

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

```
unsigned f( unsigned x, unsigned v[] ) {  
    if (x==0)  
        return 0;  
  
    unsigned m = 0;  
  
    for ( unsigned y = 1; y < x; y++ )  
        m = max( m, v[y] + f( x-y, v ) );  
  
    return m;
```

O(x)

O(x^2)

O($x \cdot v$)

Los algoritmos directos de ordenación, respecto de los indirectos:

- Presentan una menor complejidad temporal y sus tiempos de ejecución absolutos son menores.
- Presentan una mayor complejidad temporal y sus tiempos de ejecución absolutos son mayores.
- Presentan una mayor complejidad temporal si bien sus tiempos de ejecución absolutos son menores.

Si $f_1(n) \in O(g_1(n))$ y $f_2(n) \in O(g_2(n))$ entonces:

- f₁(n)+f₂(n) ∈ O(maximo(g₁(n),g₂(n)))
- f₁(n)+f₂(n) ∈ O (g₁(n)+g₂(n))
- Ambas son correctas

¿Por que se emplean funciones de coste para expresar el coste de una algoritmo?

- X Para poder expresar el coste de los algoritmos con mayor exactitud
- ✓ Para que la expresión del coste del algoritmo sea válida para cualquier entrada al mismo
- Para poder expresar el coste de un algoritmo mediante una expresión matemática

$f(n) = 5n + 3m \cdot n + 11$ entonces $f(n)$ pertenece a:

- O $O(n \cdot m)$.
- O $O(n^m)$.
- Las dos son correctas

Si dos algoritmos tienen la misma complejidad asintótica:

- Ⓛ No necesitan exactamente el mismo tiempo para su ejecución.
- Ⓜ Necesitan exactamente el mismo tiempo para su ejecución.
- Ⓝ Ninguna de las anteriores

$f(n) = 5n+5$ ¿ $f(n)$ pertenece a $O(n)$?

- Sí. El valor de c es 6 y el valor mínimo de n_0 es de 5
- Sí. El valor de c es 5 y el valor mínimo de n_0 es de 3
- Sí. El valor de c es 9 y el valor mínimo de n_0 es de 1

¿Cuál de los siguientes algoritmos de ordenación tiene menor complejidad?

- Mergesort
- Inserción directa
- Burbuja

Dado el polinomio $f(n) = A(m) n^m + A(m-1) n^{m-1} + \dots + A_0$, con $A(m) \in \mathbb{R}_+$ entonces f pertenece al orden:

- $\mathcal{O}(n^m)$.
- La dos respuestas anteriores son correctas.
- $\Omega(n^m)$.

Cual de las siguientes definiciones es cierta:

- Las cotas de complejidad se emplean cuando para una misma talla se obtienen diferentes complejidades dependiendo de la entrada al problema.
- Las cotas de complejidad se emplean cuando para diferentes tallas se obtienen diferentes complejidades dependiendo de la entrada al problema.
- Ninguna de las anteriores

Si $f(n) \in \Omega(g(n))$ entonces:

- $\exists c, n_0 \in \mathbb{R}^+: f(n) \leq c \cdot g(n) \forall n \geq n_0$
- $\exists c, n_0 \in \mathbb{R}^+: f(n) \geq c \cdot g(n) \forall n$
- $\exists c, n_0 \in \mathbb{R}^+: f(n) \geq c \cdot g(n) \forall n \geq n_0$

¿Cuál es el objetivo de la etapa de análisis en el Diseño y Análisis de un Algoritmo?

- Estimar los recursos que consumirá el algoritmo una vez implementado.
- Determinar el lenguaje y herramientas disponibles para su desarrollo.
- Estimar la potencia y características del equipo informático necesarios para el correcto funcionamiento del algoritmo.

En un algoritmo recursivo, la forma de dividir el problema en subproblemas:

- Influye en su complejidad temporal.
- No influye en ninguna de sus complejidades.
- Influye en la complejidad espacial del mismo.

Ordena de menor a mayor las siguientes complejidades:

1. $O(1)$
2. $O(n^2)$
3. $O(n \lg n)$
4. $O(n!)$

 3,1,2 y 4

 1,3,2 y 4

1,3,4 y 2

Cuando para distintas instancias de problema con el mismo tamaño no obtenemos el mismo resultado:

- Calculamos el máximo y mínimo coste que nos puede dar el algoritmo.
 - No es posible calcular la complejidad a priori y debemos ejecutar el programa varias veces con la misma talla y obtener el tiempo medio para hallar la complejidad media.
 - No se puede aplicar la técnica de paso de programa, ya que esta técnica es para calcular la complejidad a priori.

El coste asociado a la siguiente ecuación de recurrencia es:

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

- ⊗ $\Theta(n^2 \lg(n))$
- ⊗ $\Theta(n \lg(n^2))$
- ⊗ $\Theta(n \lg(n))$

El estudio de la complejidad resulta realmente interesante para tamaños grandes de problema por varios motivos:

- Ninguna de las anteriores.
- Las diferencias reales en tiempo de compilación de algoritmos con diferente coste para tamaños pequeños del problema no suelen ser muy significativas.
- Las diferencias reales en tiempo de ejecución de algoritmos con diferente coste para tamaños grandes del problema no suelen ser muy significativas.

$f(n) = 10n+7$ ¿ $f(n)$ pertenece a $O(n^2)$?

- Sí Para cualquier valor de c positivo siempre existe un n_0 a partir del que se cumple.
- Sí. Para $c = 1$ y a partir de un valor de $n_0 = 10$.
- No.

La talla o tamaño de un problema depende de:

- ⬤ Conjunto de valores asociados a la entrada del problema.
- ⬤ Conjunto de valores asociados a la salida del problema.
- ⬤ Conjunto de valores asociados a la entrada y salida del problema.

El sumatorio, desde $i=1$ hasta n , de i^k pertenece a:

- $\Theta(n^{k+1})$
- $\Theta(n^k)$
- Ninguna de las anteriores

La complejidad de la función A2 es:

```
Función A2 (n, a: entero):entero;
Var r: entero; fvar
    si ( $a^2 > n$ ) devuelve 0
    sino
        r:= A2(n,  $\lceil a \rceil$ );
        opción
             $n < a^2$ :     devuelve r;
             $n \geq a^2$ :   devuelve r + a;
        fopción
    fsi
fin
```

- O(\sqrt{n} / a)
- O(n / \sqrt{a})
- O($\sqrt{n} \cdot a$)

Si $f_1(n) \in O(g_1(n))$ y $f_2(n) \in O(g_2(n))$ entonces:

f₁(n) · f₂(n) ∈ O(maximo(g₁(n), g₂(n)))

f₁(n) · f₂(n) ∈ O(g₁(n) · g₂(n))

Ambas son correctas

¿El tiempo de ejecución de un algoritmo depende de la talla del problema?

 Sí, siempre

Nc, nunca

 Nc necesariamente

La complejidad de la función TB es:

```
función TB (A: vector[λ]; iz : de : N) : N
var n,i:N;
n=iz,de+1
opcion
  (n < 1) : devuelve (0);
  (n = 1) : devuelve (1);
  (n > 1) : si (A[iz] = A[de]) entonces
              devuelve (TB(A,iz+1,de-1)+1);
            sino
              devuelve (TB(A,iz+1,de-1));
            finsi;
opcion
fin
```

- $\Theta(n^2 \cdot \lg n)$
- $\Theta(n)$
- $\Theta(n \cdot \lg n)$

Un algoritmo cuya talla es n y que tarda 40^n segundos en resolver cualquier instancia tiene una complejidad temporal:

- $\Theta(n^n)$
- $\Theta(4^n)$
- Ninguna de las anteriores

El caso base de una ecuación de recurrencia asociada a la complejidad temporal de un algoritmo expresa:

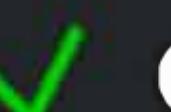
- El coste de dicho algoritmo en el mejor de los casos.
- El coste de dicho algoritmo en el peor de los casos.
- Ninguna de las anteriores

¿Cuál de las siguientes jerarquías de complejidades es la correcta?

- ... < $O(2^n)$ < $O(n!)$ < $O(n^n)$
- ... < $O(n!)$ < $O(2^n)$ < $O(n^n)$
- $O(1)$ < $O(\lg n)$ < $O(\lg \lg n)$ < ...

Se pretende resolver un problema de maximización utilizando ramificación y poda y para ello se dispone de tres posibles cotas optimistas.

¿Cuál deberíamos utilizar para tratar de reducir la cantidad de nodos a explorar?

-  La que obtenga valores más pequeños.
-  La que obtenga valores más elevados.
- La que más se acerque a una heurística voraz que obtenga una solución aproximada del problema.

Di cuál de las siguientes afirmaciones es falsa:

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

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

Respostes per a la modalitat A

1. Es vol ordenar d nombres diferents compresos entre 1 i n . Per a fer-ho s'usa un vector de n booleans que s'inicialitzen primer a *false*. A continuació es recorren els d nombres canviant els valors de l'element del vector de booleans corresponent al seu nombre a *true*. Finalment es recorre el vector de booleans escrivint els índexs dels elements del vector de booleans que són *true*. És aquest algorisme més ràpid (asimptòticament) que el *mergesort*?
 - (a) Només si $d \log d > k n$ (on k és una constant que depén de la implementació)
 - (b) Sí, ja que el *mergesort* és $O(n \log n)$ i est és $O(n)$
 - (c) No, ja que aquest algorisme ha de recórrer diverses vegades el vector de booleans.
2. En una quadrícula es vol dibuixar el contorn d'un quadrat de n caselles de costat. Quina serà la complexitat temporal del millor algorisme que pugui existir?
 - (a) $O(n^2)$
 - (b) $O(n)$
 - (c) $O(\sqrt{n})$
3. La complexitat en el millor dels casos d'un algorisme de *ramificació i poda* ...
 - (a) ... és sempre exponencial respecte del nombre de decisions a prendre.
 - (b) ... sol ser polinòmica respecte del nombre d'alternatives per cada decisió.
 - (c) ... pot ser polinòmica respecte del nombre de decisions a prendre.
4. La solució recursiva ingènua (però correcta) a un problema d'optimització crida més d'una vegada a la funció amb els mateixos paràmetres. Una de les següents tres afirmacions és falsa.
 - (a) Es pot millorar l'eficiència de l'algorisme convertint l'algorisme recursiu directament en iteratiu sense canviar el seu funcionament bàsic.
 - (b) Es pot millorar l'eficiència de l'algorisme guardant en una taula el valor retornat per a cada conjunt de paràmetres de cada trucada quan aquesta es produueix per primera vegada.
 - (c) Es pot millorar l'eficiència de l'algorisme definint per endavant l'ordre en el qual s'han de calcular les solicions als subproblemes i omplint una taula en aquest ordre.

5. Quan s'usa un algorisme voraç per abordar la resolució d'un problema d'optimització per selecció discreta (és a dir, un problema per al qual la solució consisteix a trobar un subconjunt del conjunt d'elements que optimitza una determinada funció), quina d'aquestes tres coses és impossible que ocórrrega?
- (a) Que es reconsidera la decisió ja presa anteriorment respecte a la selecció d'un element a la vista de la decisió que s'ha de prendre en l'instant actual.
 - (b) Que l'algorisme no trobe cap solució.
 - (c) Que la solució no siga l'òptima.
6. Quin dels següents algorismes proveiria una fita pessimista per al problema de trobar el camí mes curt entre dues ciutats (se suposa que el graf és connex)?
- (a) Calcular la distància recorreguda movent-se a l'atzar pel graf fins a arribar (per atzar) a la ciutat de destinació.
 - (b) Calcular la distància geomètrica (en línia recta) entre la ciutat d'origen i la de destinació.
 - (c) Per a totes les ciutats a les quals es pot arribar en un pas des de la ciutat inicial, sumar la distància a aquesta ciutat i la distància geomètrica fins a la ciutat de destinació.
7. Per a quin d'aquests problemes d'optimització existeix una solució voraç?
- (a) L'arbre de recobriment mínim per a un graf no dirigit amb pesos.
 - (b) El problema de la motxilla discreta.
 - (c) El problema de l'assignació de cost mínim de n tasques a n treballadors quan el cost d'assignar la tasca i al treballador j , c_{ij} està tabulat en una matriu.
8. Un problema de grandària n pot transformar-se en temps $O(n^2)$ en un altre de grandària $n - 1$. D'altra banda, la solució al problema quan la talla és 1 requereix un temps constant. Quina d'aquestes classes de cost temporal asymptòtic és la més ajustada?
- (a) $O(2^n)$
 - (b) $O(n^2)$
 - (c) $O(n^3)$
9. Quin d'aquests problemes té una solució eficient que usa *programació dinàmica*?
- (a) El problema de l'assignació de tasques.
 - (b) La motxilla discreta sense restriccions addicionals.
 - (c) El problema del canvi.

10. Si un problema d'optimització ho és per a una funció que pren valors contínus ...
- (a) La programació dinàmica recursiva pot resultar molt més eficient que la programació dinàmica iterativa quant a l'ús de memòria.
 - (b) La programació dinàmica iterativa sempre és molt més eficient que la programació dinàmica recursiva quant a l'ús de memòria.
 - (c) L'ús de memòria de la programació dinàmica iterativa i de la programació dinàmica recursiva és el mateix independentment de si el domini és discret o contínuo.
11. En resoldre el problema del viatjant de comerç mitjançant tornada arreplega, quina d'aquestes fites optimistes s'espera que pode millor l'arbre de cerca?
- (a) S'ordenen les arestes restants de menor a major distància i es calcula la suma de les k arestes més curtes, on k és el nombre de salts que ens queden per donar.
 - (b) Es multiplica k per la distància de l'aresta més curta que ens queda per considerar, on k és el nombre de salts que ens queden per donar.
 - (c) Es resol la resta del problema usant un algorisme voraç que afegeix cada vegada al camí el vèrtex més proper a l'últim afegit.
12. La versió de *Quicksort* que utilitza com a pivot l'element del vector que ocupa la primera posició ...
- (a) ... no presenta cas millor i pitjor per a instàncies de la mateixa grandària.
 - (b) ... es comporta pitjor quan el vector ja està ordenat.
 - (c) ... es comporta millor quan el vector ja està ordenat.

13. El programa següent resol el problema de tallar un tub de longitud n en segments de longitud entera entre 1 i n de manera que es maximitze el preu d'acord amb una taula que dóna el preu per a cada longitud, però en falta un tros. Què hauria d'anar en lloc de **XXXXXX**?

```
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(XXXXXX) );
    }
    r[n]=q;
    return q;
}
```

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

14. Si $f(n) \in O(n^3)$, pot passar que $f(n) \in O(n^2)$?

- (a) És perfectament possible, ja que $O(n^2) \subset O(n^3)$
- (b) Només per a valors baixos de n
- (c) No, perquè $n^3 \notin O(n^2)$

15. Digueu quin d'aquests tres algorismes no és un algorisme de “divideix i venceràs”

- (a) L'algorisme de Prim
- (b) Quicksort
- (c) Mergesort

16. La complexitat temporal en el millor dels casos...

- (a) ... és una funció de la grandària o talla del problema que ha d'estar definida per tots els possibles valors d'aquesta.
- (b) ... és el temps que tarda l'algorisme a resoldre el problema de grandària o talla més petita que se li pot presentar.
- (c) Les dues anteriors són certes.

17. L'ús de funcions de fita en ramificació i poda...

- (a) ... transforma en polinòmiques complexitats que abans eren exponencials.
- (b) ... pot reduir el nombre d'instàncies del problema que pertanyen al cas pitjor.
- (c) ... garanteix que l'algorisme serà més eficient davant qualsevol instància del problema.

18. Donades les funcions següents:

```
// Precondició: { 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() );
}
```

Es vol reduir la complexitat temporal de la funció g usant programació dinàmica iterativa. Quina en seria la complexitat espacial?

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

19. Quin d'aquests tres problemes d'optimització no té, o no se li'n coneix, una solució voraç (*greedy*) que siga òptima?

- (a) El problema de la motxilla contínua o amb fraccionament.
- (b) El problema de la motxilla discreta.
- (c) L'arbre de cobertura de cost mínim d'un graf connex.

20. En els algorismes de *ramificació i poda* ...

- (a) Una fita optimista és necessàriament un valor insuperable; si no fóra així, es podria podar el node que condueix a la solució òptima.
- (b) Una fita optimista és necessàriament un valor abastable; si no és així, no està garantit que es trobe la solució òptima.
- (c) Una fita pessimista és el valor que com a màxim aconsegueix qualsevol node factible que no és l'òptim.

21. Una d'aquestes tres situacions no és possible:

- (a) $f(n) \in \Omega(n^2)$ i $f(n) \in O(n)$
- (b) $f(n) \in O(n)$ i $f(n) \in \Omega(1)$
- (c) $f(n) \in O(n)$ i $f(n) \in O(n^2)$

22. En els algorismes de *ramificació i poda*, el valor d'una fita pessimista és més gran que el valor d'una fita optimista? (s'entén que ambdues fites s'apliquen sobre el mateix node)
- (a) No, mai no és així.
 - (b) En general sí, si es tracta d'un problema de maximització, encara que en ocasions tots dos valors poden coincidir.
 - (c) En general sí, si es tracta d'un problema de minimització, encara que en ocasions tots dos valors poden coincidir.
23. Un d'aquests tres problemes no té una solució eficient que seguisca l'esquema de programació dinàmica
- (a) El problema de les torres d'Hanoi
 - (b) El problema de la motxilla discreta.
 - (c) El problema de tallar un tub de longitud n en segments de longitud sencera entre 1 i n de manera que es maximitze el preu d'accord amb una taula que dóna el preu per a cada longitud.
24. Donat un problema d'optimització, el mètode voraç...
- (a) ... sempre obté la solució òptima.
 - (b) ... sempre obté una solució factible.
 - (c) ... garanteix la solució òptima només para determinats problemes.
25. Donat un problema d'optimització qualsevol, l'estrategia de *tornada arrengonada* garanteix la solució òptima?
- (a) Sí, ja que aquest mètode analitza totes les possibilitats.
 - (b) És condició necessària que el domini de les decisions siga discret o discretizable i que el nombre de decisions a prendre estiga fitat.
 - (c) Sí, sempre que el domini de les decisions siga discret o discretizable i a més s'usen mecanismes de poda basats en la millor solució fins al moment.
26. Garanteix l'ús d'una estratègia “divideix i venceràs” l'existència d'una solució de complexitat temporal polinòmica a qualsevol problema?
- (a) No
 - (b) Sí, en qualsevol cas.
 - (c) Sí, però sempre que la complexitat temporal conjunta de les operacions de descomposició del problema i la combinació de les solucions siga polinòmica.

27. Si per resoldre un mateix problema usem un algorisme de tornada arrere i el modifiquem mínimament per convertir-ho en un algorisme de ramificació i poda, què canviem realment?
- (a) L'algorisme pot aprofitar millor les fites optimistes.
(b) Canviem la funció que donem a la fita pessimista.
(c) La comprovació de les solicions factibles: en ramificació i poda no és necessari ja que només genera nodes factibles.
28. Es vol trobar el camí mes curt entre dues ciutats. Per a açò es disposa d'una taula amb la distància entre els parells de ciutats entre les quals hi ha carretera o un valor sentinella (per exemple, -1) si no n'hi ha; per això, per anar de la ciutat inicial a la final és possible que calga passar per diverses ciutats. També es coneixen les coordenades geogràfiques de cada ciutat i per tant la distància geomètrica (en línia recta) entre cada parell de ciutats. Es pretén accelerar la recerca d'un algorisme de *ramificació i poda* prioritzant els nodes vius (ciutats) que estiguin a menor distància geogràfica de la ciutat objectiu.
- (a) El nou algorisme sempre sera més ràpid.
(b) El nou algorisme no garanteix que vaja a ser més ràpid per a totes les instàncies del problema possibles.
(c) Aquesta estratègia no assegura que s'obtinga el camí mes curt.
29. La millora que en general aporta la programació dinàmica enfront de la solució ingènua s'aconsegueix gràcies al fet que ...
- (a) ... en la solució ingènua es resol moltes vegades un nombre relativament reduït de subproblemes diferents.
(b) ... en la solució ingènua es resol poques vegades un nombre relativament gran de subproblemes diferents.
(c) El nombre de vegades que es resolen els subproblemes no té res a veure amb l'eficiència dels problemes resolts mitjançant programació dinàmica.
30. La millor solució que es coneix per al problema de la motxilla contínua segueix l'esquema ...
- (a) ... *voraç*.
(b) ... *divideix i venceràs*.
(c) ... *ramificació i poda*.
31. Un algorisme recursiu basat en l'esquema *divideix i venceràs* ...
- (a) ... serà més eficient com més equitativa siga la divisió en subproblemes.
(b) ... no tindrà mai una complexitat exponencial.
(c) Les dues anteriors són certes.

32. En l'esquema de tornada arrere, els mecanismes de poda basats en la millor solució fins al moment...

- (a) ... poden eliminar solucions parcials que són factibles.
- (b) ... garanteixen que no s'explorarà mai tot l'espai de solucions possibles.
- (c) Les dues anteriors són veritables.

33. Digueu quina d'aquestes tres és la fita pessimista més ajustada al valor òptim de la motxilla discreta:

- (a) El valor de la motxilla contínua corresponent
- (b) El valor de la motxilla discreta que s'obté usant un algorisme voraç basat en el valor específic dels objectes
- (c) El valor d'una motxilla que conté tots els objectes encara que es passe del pes màxim permès

34. Siga la relació següent de recurrència

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

Si $T(n) \in O(n^2)$, en quin d'aquests tres casos ens podem trobar?

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

35. Quan la descomposició recursiva d'un problema dóna lloc a subproblems de grandària similar, quin esquema promet ser més apropiat?

- (a) Divideix i venceràs, sempre que es garantísca que els subproblems no són de la mateixa grandària.
- (b) Programació dinàmica.
- (c) El mètode voraç

36. En l'esquema de *tornada arrere* l'ordre en el qual es van assignant els diferents valors a les components del vector que contindrà la solució...

- (a) ... és irrelevant si no s'utilitzen mecanismes de poda basats en la millor solució fins al moment.
- (b) ... pot ser rellevant si s'utilitzen mecanismes de poda basats en estimacions optimistes.
- (c) Les dues anteriors són certes.

37. Quan es resol el problema de la motxilla discreta usant l'estrategia de tornada arrere, pot ocórrer que es tarde menys a trobar la solució òptima si es prova primer a ficar-hi cada objecte abans de no ficar-l'hi?
- (a) Sí, però només si s'usen fites optimistes per podar l'arbre de recerca.
 - (b) Sí, tant si s'usen fites optimistes per podar l'arbre de recerca com si no.
 - (c) No, ja que en qualsevol cas s'han d'explorar totes les solicions factibles.
38. En un problema d'optimització, si el domini de les decisions és un conjunt infinit,
- (a) és probable que a través de programació dinàmica s'obtinga un algorisme eficaç que el resolga.
 - (b) podrem aplicar l'esquema de tornada arrere sempre que es tracte d'un conjunt infinit numerable.
 - (c) una estratègia voraç pot ser l'única alternativa.
39. El valor que s'obté amb el mètode voraç per al problema de la motxilla discreta és ...
- (a) ... una fita inferior per al valor òptim, però que mai coincideix amb aquest.
 - (b) ... una fita inferior per al valor òptim que de vegades pot ser igual a aquest.
 - (c) ... una fita superior per al valor òptim.
40. Siga A una matriu quadrada $n \times n$. Es tracta de buscar una permutació de les columnes tal que la suma dels elements de la diagonal de la matriu resultant siga mínima. Indiqueu quina de les següents afirmacions és falsa.
- (a) La complexitat temporal de la millor solució possible al problema és $O(n^2)$.
 - (b) La complexitat temporal de la millor solució possible al problema és $O(n!)$.
 - (c) Si es construeix una solució al problema basada en l'esquema de ramificació i poda, una bona elecció de fites optimistes i pessimistes podria evitar l'exploració de totes les permutacions possibles.

Respostes per a la modalitat B

1. Si $f(n) \in O(n^3)$, pot passar que $f(n) \in O(n^2)$?
 - (a) És perfectament possible, ja que $O(n^2) \subset O(n^3)$
 - (b) Només per a valors baixos de n
 - (c) No, perquè $n^3 \notin O(n^2)$
2. Donat un problema d'optimització qualsevol, l'estrategia de *tornada arrenglada* garanteix la solució òptima?
 - (a) Sí, ja que aquest mètode analitza totes les possibilitats.
 - (b) És condició necessària que el domini de les decisions siga discret o discretizable i que el nombre de decisions a prendre estiga fitat.
 - (c) Sí, sempre que el domini de les decisions siga discret o discretizable i a més s'usen mecanismes de poda basats en la millor solució fins al moment.
3. El valor que s'obté amb el mètode voraç per al problema de la motxilla discreta és ...
 - (a) ... una fita inferior per al valor òptim, però que mai coincideix amb aquest.
 - (b) ... una fita inferior per al valor òptim que de vegades pot ser igual a aquest.
 - (c) ... una fita superior per al valor òptim.
4. La complexitat en el millor dels casos d'un algorisme de *ramificació i poda* ...
 - (a) ... és sempre exponencial respecte del nombre de decisions a prendre.
 - (b) ... sol ser polinòmica respecte del nombre d'alternatives per cada decisió.
 - (c) ... pot ser polinòmica respecte del nombre de decisions a prendre.
5. Quan la descomposició recursiva d'un problema dóna lloc a subproblems de grandària similar, quin esquema promet ser més apropiat?
 - (a) Divideix i venceràs, sempre que es garantísca que els subproblems no són de la mateixa grandària.
 - (b) Programació dinàmica.
 - (c) El mètode voraç
6. Digueu quin d'aquests tres algorismes no és un algorisme de "divideix i venceràs"
 - (a) L'algorisme de Prim
 - (b) Quicksort
 - (c) Mergesort

7. Un algorisme recursiu basat en l'esquema *divideix i venceràs* ...
- (a) ... serà més eficient com més equitativa siga la divisió en subproblemes.
- (b) ... no tindrà mai una complexitat exponencial.
- (c) Les dues anteriors són certes.
8. Si per resoldre un mateix problema usem un algorisme de tornada arrere i el modifiquem mínimament per convertir-ho en un algorisme de ramificació i poda, què canviem realment?
- (a) L'algorisme pot aprofitar millor les fites optimistes.
- (b) Canviem la funció que donem a la fita pessimista.
- (c) La comprovació de les solucions factibles: en ramificació i poda no és necessari ja que només genera nodes factibles.
9. En l'esquema de *tornada arrere* l'ordre en el qual es van assignant els diferents valors a les components del vector que contindrà la solució...
- (a) ... és irrelevat si no s'utilitzen mecanismes de poda basats en la millor solució fins al moment.
- (b) ... pot ser rellevant si s'utilitzen mecanismes de poda basats en estimacions optimistes.
- (c) Les dues anteriors són certes.
10. L'ús de funcions de fita en ramificació i poda...
- (a) ... transforma en polinòmiques complexitats que abans eren exponencials.
- (b) ... pot reduir el nombre d'instàncies del problema que pertanyen al cas pitjor.
- (c) ... garanteix que l'algorisme serà més eficient davant qualsevol instància del problema.
11. La complexitat temporal en el millor dels casos...
- (a) ... és una funció de la grandària o talla del problema que ha d'estar definida per tots els possibles valors d'aquesta.
- (b) ... és el temps que tarda l'algorisme a resoldre el problema de grandària o talla més petita que se li pot presentar.
- (c) Les dues anteriors són certes.

12. Es vol trobar el camí mes curt entre dues ciutats. Per a açò es disposa d'una taula amb la distància entre els parells de ciutats entre les quals hi ha carretera o un valor sentinella (per exemple, -1) si no n'hi ha; per això, per anar de la ciutat inicial a la final és possible que calga passar per diverses ciutats. També es coneixen les coordenades geogràfiques de cada ciutat i per tant la distància geomètrica (en línia recta) entre cada parell de ciutats. Es pretén accelerar la recerca d'un algorisme de *ramificació i poda* prioritant els nodes vius (ciutats) que estiguin a menor distància geogràfica de la ciutat objectiu.
- (a) El nou algorisme sempre sera més ràpid.
 - (b) El nou algorisme no garanteix que vaja a ser més ràpid per a totes les instàncies del problema possibles.
 - (c) Aquesta estratègia no assegura que s'obtinga el camí mes curt.
13. Quin d'aquests tres problemes d'optimització té, o no se li'n coneix, una solució voraç (*greedy*) que siga òptima?
- (a) El problema de la motxilla contínua o amb fraccionament.
 - (b) El problema de la motxilla discreta.
 - (c) L'arbre de cobertura de cost mínim d'un graf connex.
14. Quan s'usa un algorisme voraç per abordar la resolució d'un problema d'optimització per selecció discreta (és a dir, un problema per al qual la solució consisteix a trobar un subconjunt del conjunt d'elements que optimitza una determinada funció), quina d'aquestes tres coses és impossible que ocórrrega?
- (a) Que es reconsidera la decisió ja presa anteriorment respecte a la selecció d'un element a la vista de la decisió que s'ha de prendre en l'instant actual.
 - (b) Que l'algorisme no trobe cap solució.
 - (c) Que la solució no siga l'òptima.
15. Quin d'aquests problemes té una solució eficient que usa *programació dinàmica*?
- (a) El problema de l'assignació de tasques.
 - (b) La motxilla discreta sense restriccions addicionals.
 - (c) El problema del canvi.
16. En els algorismes de *ramificació i poda*, el valor d'una fita pessimista és més gran que el valor d'una fita optimista? (s'entén que ambdues fites s'apliquen sobre el mateix node)
- (a) No, mai no és així.
 - (b) En general sí, si es tracta d'un problema de maximització, encara que en ocasions tots dos valors poden coincidir.
 - (c) En general sí, si es tracta d'un problema de minimització, encara que en ocasions tots dos valors poden coincidir.

17. Garanteix l'ús d'una estratègia "divideix i venceràs" l'existència d'una solució de complexitat temporal polinòmica a qualsevol problema?

- (a) No
- (b) Sí, en qualsevol cas.
- (c) Sí, però sempre que la complexitat temporal conjunta de les operacions de descomposició del problema i la combinació de les solucions siga polinòmica.

18. El programa següent resol el problema de tallar un tub de longitud n en segments de longitud entera entre 1 i n de manera que es maximitzze el preu d'acord amb una taula que dóna el preu per a cada longitud, però en falta un tros. Què hauria d'anar en lloc de **XXXXXX**?

```
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(XXXXXX));  
        }  
        r[n]=q;  
        return q;  
    }  
}
```

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

19. Donat un problema d'optimització, el mètode voraç...

- (a) ... sempre obté la solució òptima.
- (b) ... sempre obté una solució factible.
- (c) ... garanteix la solució òptima només para determinats problemes.

20. En un problema d'optimització, si el domini de les decisions és un conjunt infinit,

- (a) és probable que a través de programació dinàmica s'obtinga un algorisme eficaç que el resolga.
- (b) podrem aplicar l'esquema de tornada arrere sempre que es tracte d'un conjunt infinit numerable.
- (c) una estratègia voraç pot ser l'única alternativa.

21. Un d'aquests tres problemes no té una solució eficient que seguisca l'esquema de programació dinàmica
- (a) El problema de les torres d'Hanoi
 - (b) El problema de la motxilla discreta.
 - (c) El problema de tallar un tub de longitud n en segments de longitud sencera entre 1 i n de manera que es maximitze el preu d'acord amb una taula que dóna el preu per a cada longitud.
22. La millora que en general aporta la programació dinàmica enfront de la solució ingènua s'aconsegueix gràcies al fet que ...
- (a) ... en la solució ingènua es resol moltes vegades un nombre relativament reduït de subproblemes diferents.
 - (b) ... en la solució ingènua es resol poques vegades un nombre relativament gran de subproblemes diferents.
 - (c) El nombre de vegades que es resolen els subproblemes no té res a veure amb l'eficiència dels problemes resolts mitjançant programació dinàmica.
23. Si un problema d'optimització ho és per a una funció que pren valors contínus ...
- (a) La programació dinàmica recursiva pot resultar molt més eficient que la programació dinàmica iterativa quant a l'ús de memòria.
 - (b) La programació dinàmica iterativa sempre és molt més eficient que la programació dinàmica iterativa quant a l'ús de memòria.
 - (c) L'ús de memòria de la programació dinàmica iterativa i de la programació dinàmica recursiva és el mateix independentment de si el domini és discret o continu.
24. Quin dels següents algorismes proveiria una fita pessimista per al problema de trobar el camí mes curt entre dues ciutats (se suposa que el graf és connex)?
- (a) Calcular la distància recorreguda movent-se a l'atzar pel graf fins a arribar (per atzar) a la ciutat de destinació.
 - (b) Calcular la distància geomètrica (en línia recta) entre la ciutat d'origen i la de destinació.
 - (c) Per a totes les ciutats a les quals es pot arribar en un pas des de la ciutat inicial, sumar la distància a aquesta ciutat i la distància geomètrica fins a la ciutat de destinació.

25. Donades les funcions següents:

```
// Precondició: { 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() );
}
```

Es vol reduir la complexitat temporal de la funció g usant programació dinàmica iterativa. Quina en seria la complexitat espacial?

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

26. Una d'aquestes tres situacions no és possible:

- (a) $f(n) \in \Omega(n^2)$ i $f(n) \in O(n)$
- (b) $f(n) \in O(n)$ i $f(n) \in \Omega(1)$
- (c) $f(n) \in O(n)$ i $f(n) \in O(n^2)$

27. Siga A una matriu quadrada $n \times n$. Es tracta de buscar una permutació de les columnes tal que la suma dels elements de la diagonal de la matriu resultant siga mínima. Indiqueu quina de les següents afirmacions és falsa.

- (a) La complexitat temporal de la millor solució possible al problema és $O(n^2)$.
- (b) La complexitat temporal de la millor solució possible al problema és $O(n!)$.
- (c) Si es construeix una solució al problema basada en l'esquema de ramificació i poda, una bona elecció de fites optimistes i pessimistes podria evitar l'exploració de totes les permutacions possibles.

28. La versió de *Quicksort* que utilitza com a pivot l'element del vector que ocupa la primera posició ...

- (a) ... no presenta cas millor i pitjor per a instàncies de la mateixa grandària.
- (b) ... es comporta pitjor quan el vector ja està ordenat.
- (c) ... es comporta millor quan el vector ja està ordenat.

29. Un problema de grandària n pot transformar-se en temps $O(n^2)$ en un altre de grandària $n - 1$. D'altra banda, la solució al problema quan la talla és 1 requereix un temps constant. Quina d'aquestes classes de cost temporal asymptòtic és la més ajustada?
- (a) $O(2^n)$
 - (b) $O(n^2)$
 - (c) $O(n^3)$
30. En els algorismes de *ramificació i poda* ...
- (a) Una fita optimista és necessàriament un valor insuperable; si no fóra així, es podria podar el node que conduceix a la solució òptima.
 - (b) Una fita optimista és necessàriament un valor abastable; si no és així, no està garantit que es trobe la solució òptima.
 - (c) Una fita pessimista és el valor que com a màxim aconsegueix qualsevol node factible que no és l'òptim.
31. En l'esquema de tornada arrere, els mecanismes de poda basats en la millor solució fins al moment...
- (a) ... poden eliminar solucions parcials que són factibles.
 - (b) ... garanteixen que no s'explorarà mai tot l'espai de solucions possibles.
 - (c) Les dues anteriors són veritables.
32. La solució recursiva ingènua (però correcta) a un problema d'optimització crida més d'una vegada a la funció amb els mateixos paràmetres. Una de les següents tres afirmacions és falsa.
- (a) Es pot millorar l'eficiència de l'algorisme convertint l'algorisme recursiu directament en iteratiu sense canviar el seu funcionament bàsic.
 - (b) Es pot millorar l'eficiència de l'algorisme guardant en una taula el valor retornat per a cada conjunt de paràmetres de cada trucada quan aquesta es produueix per primera vegada.
 - (c) Es pot millorar l'eficiència de l'algorisme definint per endavant l'ordre en el qual s'han de calcular les solucions als subproblemes i omplint una taula en aquest ordre.
33. La millor solució que es coneix per al problema de la motxilla contínua segueix l'esquema ...
- (a) ... *voraç*.
 - (b) ... *divideix i venceràs*.
 - (c) ... *ramificació i poda*.

34. Per a quin d'aquests problemes d'optimització existeix una solució voraç?
- (a) L'arbre de recobriment mínim per a un graf no dirigit amb pesos.
- (b) El problema de la motxilla discreta.
- (c) El problema de l'assignació de cost mínim de n tasques a n treballadors quan el cost d'assignar la tasca i al treballador j , c_{ij} està tabulat en una matriu.
35. En resoldre el problema del viatjant de comerç mitjançant tornada arre, quina d'aquestes fites optimistes s'espera que pode millor l'arbre de cerca?
- (a) S'ordenen les arestes restants de menor a major distància i es calcula la suma de les k arestes més curtes, on k és el nombre de salts que ens queden per donar.
- (b) Es multiplica k per la distància de l'aresta més curta que ens queda per considerar, on k és el nombre de salts que ens queden per donar.
- (c) Es resol la resta del problema usant un algorisme voraç que afegeix cada vegada al camí el vèrtex més proper a l'últim afegit.
36. Digueu quina d'aquestes tres és la fita pessimista més ajustada al valor òptim de la motxilla discreta:
- (a) El valor de la motxilla contínua corresponent
- (b) El valor de la motxilla discreta que s'obté usant un algorisme voraç basat en el valor específic dels objectes
- (c) El valor d'una motxilla que conté tots els objectes encara que es passe del pes màxim permés
37. Quan es resol el problema de la motxilla discreta usant l'estrategia de tornada arrere, pot ocórrer que es tarde menys a trobar la solució òptima si es prova primer a ficar-hi cada objecte abans de no ficar-l'hi?
- (a) Sí, però només si s'usen fites optimistes per podar l'arbre de cerca.
- (b) Sí, tant si s'usen fites optimistes per podar l'arbre de cerca com si no.
- (c) No, ja que en qualsevol cas s'han d'explorar totes les solucions factibles.
38. En una quadrícula es vol dibuixar el contorn d'un quadrat de n caselles de costat. Quina serà la complexitat temporal del millor algorisme que pugui existir?
- (a) $O(n^2)$
- (b) $O(n)$
- (c) $O(\sqrt{n})$

39. Es vol ordenar d nombres diferents compresos entre 1 i n . Per a fer-ho s'usa un vector de n booleans que s'inicialitzen primer a *false*. A continuació es recorren els d nombres canviant els valors de l'element del vector de booleans corresponent al seu nombre a *true*. Finalment es recorre el vector de booleans escrivint els índexs dels elements del vector de booleans que són *true*. És aquest algorisme més ràpid (asimptòticament) que el *mergesort*?

- (a) Només si $d \log d > k n$ (on k és una constant que depén de la implementació)
- (b) Sí, ja que el *mergesort* és $O(n \log n)$ i est és és $O(n)$
- (c) No, ja que aquest algorisme ha de recórrer diverses vegades el vector de booleans.

40. Siga la relació següent de recurrència

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

Si $T(n) \in O(n^2)$, en quin d'aquests tres casos ens podem trobar?

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

Respuestas para la modalidad C

1. 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 recorrida moviéndose al azar por el grafo hasta llegar (por azar) a la ciudad destino.
- (b) Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.
- (c) 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.

2. 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(XXXXXXXX));  
    }  
    r[n]=q;  
    return q;  
}
```

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

3. Sea la siguiente relación de recurrencia

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

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

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

4. Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

- (a) Se 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.
- (b) 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.
- (c) Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.

5. Si un problema de optimización lo es para una función que toma valores continuos ...

- (a) La programación dinámica recursiva puede resultar mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.
- (b) La programación dinámica iterativa siempre es mucho más eficiente que la programación dinámica recursiva 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.

6. 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) Sólo si $d \log d > k n$ (donde k es una constante que depende de la implementación)
- (b) Sí, ya que el *mergesort* es $O(n \log n)$ y este es $O(n)$
- (c) No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.

7. 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) $O(\sqrt{n})$
8. El uso de funciones de cota en ramificación y poda...
- (a) ... transforma en polinómicas complejidades que antes eran exponenciales.
 - (b) ... puede reducir el número de instancias del problema que pertenecen al caso peor.
 - (c) ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.
9. 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 dos anteriores son ciertas.
10. Garantiza el uso de una estrategia “divide y vencerás” la existencia de una solución de complejidad temporal polinómica a cualquier problema?
- (a) No
 - (b) Sí, en cualquier caso.
 - (c) Sí, pero siempre que la complejidad temporal conjunta de las operaciones de descomposición del problema y la combinación de las soluciones sea polinómica.
11. 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 convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.
 - (b) Se puede mejorar la eficiencia del algoritmo guardando en una tabla el valor devuelto para cada conjunto de parámetros de cada llamada cuando ésta se produce por primera vez.
 - (c) Se puede mejorar la eficiencia del algoritmo definiendo de antemano el orden en el que se deben calcular las soluciones a los subproblemas y llenando una tabla en ese orden.

12. 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 la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos
 - (c) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido
13. En los algoritmos de *ramificación y poda* ...
- (a) Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.
 - (b) Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.
 - (c) Una cota pesimista es el valor que a lo sumo alcanza cualquier nodo factible que no es el óptimo.
14. 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) ... suele ser polinómica con el número de alternativas por cada decisión.
 - (c) ... puede ser polinómica con el número de decisiones a tomar.
15. Una de estas tres situaciones no es posible:
- (a) $f(n) \in \Omega(n^2)$ y $f(n) \in O(n)$
 - (b) $f(n) \in O(n)$ y $f(n) \in \Omega(1)$
 - (c) $f(n) \in O(n)$ y $f(n) \in O(n^2)$
16. 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) El algoritmo puede aprovechar mejor las cotas optimistas.
 - (b) Cambiamos la función que damos a la cota pesimista.
 - (c) La comprobación de las soluciones factibles: en ramificación y poda no es necesario puesto que sólo genera nodos factibles.
17. Un algoritmo recursivo basado en el esquema *divide y vencerás* ...
- (a) ... será más eficiente cuanto más equitativa sea la división en subproblemas.
 - (b) ... nunca tendrá una complejidad exponencial.
 - (c) Las dos anteriores son verdaderas.

18. Dado un problema de optimización, el método voraz...
- (a) ... siempre obtiene la solución óptima.
 - (b) ... siempre obtiene una solución factible.
 - (c) ... garantiza la solución óptima sólo para determinados problemas.
19. 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 muchas veces un número relativamente pequeño de subproblemas distintos.
 - (b) ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.
 - (c) El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.
20. 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 peor cuando el vector ya está ordenado.
 - (c) ... se comporta mejor cuando el vector ya está ordenado.
21. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...
- (a) ... pueden eliminar soluciones parciales que son factibles.
 - (b) ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
 - (c) Las dos anteriores son verdaderas.
22. 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 inferior para el valor óptimo que a veces puede ser igual a este.
 - (c) ... una cota superior para el valor óptimo.

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

24. 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) cuadrática
- (b) cúbica
- (c) exponencial

25. ¿Para cuál de estos problemas de optimización existe una solución voraz?

- (a) El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
- (b) El problema de la mochila discreta.
- (c) El problema de la asignación de coste mínimo de n tareas a n trabajadores cuando el coste de asignar la tarea i al trabajador j , c_{ij} está tabulado en una matriz.

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

27. ¿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.
28. 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) $O(n^2)$
 - (c) $O(n^3)$
29. Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?
- (a) Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
 - (b) Programación dinámica.
 - (c) El método voraz
30. 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, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.
 - (b) Sí, tanto si se usan cotas optimistas para podar el árbol de búsqueda como si no.
 - (c) No, ya que en cualquier caso se deben explorar todas las soluciones factibles.
31. Si $f(n) \in O(n^3)$, ¿puede pasar que $f(n) \in O(n^2)$?
- (a) Es perfectamente posible, ya que $O(n^2) \subset O(n^3)$
 - (b) Sólo para valores bajos de n
 - (c) No, porque $n^3 \notin O(n^2)$

32. 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 $O(n^2)$.
 - (b) La complejidad temporal de la mejor solución posible al problema es $O(n!)$.
 - (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.
33. En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,
- (a) es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione.
 - (b) podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.
 - (c) una estrategia voraz puede ser la única alternativa.
34. La complejidad temporal en el mejor de los casos...
- (a) ... es una función del tamaño o talla del problema que tiene que estar definida para todos los posibles valores de ésta.
 - (b) ... es el tiempo que tarda el algoritmo en resolver el problema de tamaño o talla más pequeña que se le puede presentar.
 - (c) Las dos anteriores son verdaderas.
35. Uno de estos tres problemas no tiene una solución eficiente que siga el esquema de programación dinámica
- (a) El problema de las torres de Hanoi
 - (b) El problema de la mochila discreta.
 - (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.
36. Di cuál de estos tres algoritmos no es un algoritmo de “divide y vencerás”
- (a) El algoritmo de Prim
 - (b) Quicksort
 - (c) Mergesort

37. 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 maximización, aunque en ocasiones ambos valores pueden coincidir.
 - (c) En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
38. 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 siempre sera más rápido.
 - (b) El nuevo algoritmo no garantiza que vaya a ser más rápido para todas las instancias del problema posibles.
 - (c) Esta estrategia no asegura que se obtenga el camino mas corto.
39. 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) Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.
 - (c) 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.
40. La mejor solución que se conoce para el problema de la mochila continua sigue el esquema ...
- (a) ... voraz.
 - (b) ... divide y vencerás.
 - (c) ... ramificación y poda.

Respuestas para la modalidad D

1. El uso de funciones de cota en ramificación y poda...
 - (a) ... transforma en polinómicas complejidades que antes eran exponenciales.
 - (b) ... puede reducir el número de instancias del problema que pertenecen al caso peor.
 - (c) ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.
2. 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, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.
 - (b) Sí, tanto si se usan cotas optimistas para podar el árbol de búsqueda como si no.
 - (c) No, ya que en cualquier caso se deben explorar todas las soluciones factibles.
3. 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) $O(\sqrt{n})$
4. 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) ... suele ser polinómica con el número de alternativas por cada decisión.
 - (c) ... puede ser polinómica con el número de decisiones a tomar.
5. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...
 - (a) ... pueden eliminar soluciones parciales que son factibles.
 - (b) ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
 - (c) Las dos anteriores son verdaderas.
6. Di cuál de estos tres algoritmos no es un algoritmo de “divide y vencerás”
 - (a) El algoritmo de Prim
 - (b) Quicksort
 - (c) Mergesort

7. 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) cuadrática
- (b) cúbica
- (c) exponencial

8. 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) Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.
- (c) 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.

9. ¿Para cuál de estos problemas de optimización existe una solución voraz?

- (a) El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
- (b) El problema de la mochila discreta.
- (c) El problema de la asignación de coste mínimo de n tareas a n trabajadores cuando el coste de asignar la tarea i al trabajador j , c_{ij} está tabulado en una matriz.

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

```

void fill(price 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(XXXXXXXX) );
    }
    r[n]=q;
    return q;
}

```

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

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

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

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

13. 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 peor cuando el vector ya está ordenado.
- (c) ... se comporta mejor cuando el vector ya está ordenado.

14. 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 inferior para el valor óptimo que a veces puede ser igual a este.
 - (c) ... una cota superior para el valor óptimo.
15. La mejor solución que se conoce para el problema de la mochila continua sigue el esquema ...
- (a) ... voraz.
 - (b) ... divide y vencerás.
 - (c) ... ramificación y poda.
16. La complejidad temporal en el mejor de los casos...
- (a) ... es una función del tamaño o talla del problema que tiene que estar definida para todos los posibles valores de ésta.
 - (b) ... es el tiempo que tarda el algoritmo en resolver el problema de tamaño o talla más pequeña que se le puede presentar.
 - (c) Las dos anteriores son verdaderas.
17. 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 $O(n^2)$.
 - (b) La complejidad temporal de la mejor solución posible al problema es $O(n!)$.
 - (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.
18. 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 maximización, aunque en ocasiones ambos valores pueden coincidir.
 - (c) En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.

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

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

(b) Programación dinámica.

- (c) El método voraz

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

- (b) Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.

- (c) 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.

21. Sea la siguiente relación de recurrencia

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

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

- (a) $g(n) = n$

- (b) $g(n) = 1$

- (c) $g(n) = n^2$

22. 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 muchas veces un número relativamente pequeño de subproblemas distintos.

- (b) ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.

- (c) El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.

23. Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?
- (a) Se 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.
 - (b) 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.
 - (c) Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
24. En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,
- (a) es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione.
 - (b) podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.
 - (c) una estrategia voraz puede ser la única alternativa.
25. 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 convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.
 - (b) Se puede mejorar la eficiencia del algoritmo guardando en una tabla el valor devuelto para cada conjunto de parámetros de cada llamada cuando ésta se produce por primera vez.
 - (c) Se puede mejorar la eficiencia del algoritmo definiendo de antemano el orden en el que se deben calcular las soluciones a los subproblemas y llenando una tabla en ese orden.
26. Si $f(n) \in O(n^3)$, ¿puede pasar que $f(n) \in O(n^2)$?
- (a) Es perfectamente posible, ya que $O(n^2) \subset O(n^3)$
 - (b) Sólo para valores bajos de n
 - (c) No, porque $n^3 \notin O(n^2)$
27. 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 dos anteriores son ciertas.

28. Un algoritmo recursivo basado en el esquema *divide y vencerás* ...
- (a) ...será más eficiente cuanto más equitativa sea la división en subproblemas.
 - (b) ...nunca tendrá una complejidad exponencial.
 - (c) Las dos anteriores son verdaderas.
29. 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 siempre sera más rápido.
 - (b) El nuevo algoritmo no garantiza que vaya a ser más rápido para todas las instancias del problema posibles.
 - (c) Esta estrategia no asegura que se obtenga el camino mas corto.
30. En los algoritmos de *ramificación y poda* ...
- (a) Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.
 - (b) Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.
 - (c) Una cota pesimista es el valor que a lo sumo alcanza cualquier nodo factible que no es el óptimo.
31. 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 la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos
 - (c) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido
32. ¿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 continua o con fraccionamiento.
 - (b) El problema de la mochila discreta.
 - (c) El árbol de cobertura de coste mínimo de un grafo conexo.

33. Uno de estos tres problemas no tiene una solución eficiente que siga el esquema de programación dinámica
- (a) El problema de las torres de Hanoi
 - (b) El problema de la mochila discreta.
 - (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.
34. Garantiza el uso de una estrategia “divide y vencerás” la existencia de una solución de complejidad temporal polinómica a cualquier problema?
- (a) No
 - (b) Sí, en cualquier caso.
 - (c) Sí, pero siempre que la complejidad temporal conjunta de las operaciones de descomposición del problema y la combinación de las soluciones sea polinómica.
35. 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 se reconsidera la decisión ya tomada anteriormente respecto a la selección de un elemento a la vista de la decisión que se debe tomar en el instante actual.
 - (b) Que el algoritmo no encuentre ninguna solución.
 - (c) Que la solución no sea la óptima.
36. ¿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.
37. Una de estas tres situaciones no es posible:
- (a) $f(n) \in \Omega(n^2)$ y $f(n) \in O(n)$
 - (b) $f(n) \in O(n)$ y $f(n) \in \Omega(1)$
 - (c) $f(n) \in O(n)$ y $f(n) \in O(n^2)$
38. 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) $O(n^2)$
 - (c) $O(n^3)$

39. 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) Sólo si $d \log d > k n$ (donde k es una constante que depende de la implementación)
 - (b) Sí, ya que el *mergesort* es $O(n \log n)$ y este es $O(n)$
 - (c) No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.
40. Si un problema de optimización lo es para una función que toma valores continuos ...
- (a) La programación dinámica recursiva puede resultar mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.
 - (b) La programación dinámica iterativa siempre es mucho más eficiente que la programación dinámica recursiva 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.

Respuestas para la modalidad E

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 recursiva en cuanto al uso de memoria.
 - (b) La programación dinámica recursiva puede resultar mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.
 - (c) El uso de memoria de la programación dinámica iterativa y de la programación dinámica recursiva es el mismo independientemente de si el dominio es discreto o continuo.

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

9. La complejidad temporal en el mejor de los casos...
- (a) ... es el tiempo que tarda el algoritmo en resolver el problema de tamaño o talla más pequeña que se le puede presentar.
 - (b) Las otras dos opciones son ciertas.
 - (c) ... es una función del tamaño o talla del problema que tiene que estar definida para todos los posibles valores de ésta.
10. La mejor solución que se conoce para el problema de la mochila continua sigue el esquema ...
- (a) ... divide y vencerás.
 - (b) ... ramificación y poda.
 - (c) ... voraz.
11. La complejidad en el mejor de los casos de un algoritmo de *ramificación y poda* ...
- (a) ... es siempre exponencial con el número de decisiones a tomar.
 - (b) ... puede ser polinómica con el número de decisiones a tomar.
 - (c) ... suele ser polinómica con el número de alternativas por cada decisión.
12. Cuando se usa un algoritmo voraz para abordar la resolución de un problema de optimización por selección discreta (es decir, un problema para el cual la solución consiste en encontrar un subconjunto del conjunto de elementos que optimiza una determinada función), ¿cuál de estas tres cosas es imposible que ocurra?
- (a) Que el algoritmo no encuentre ninguna solución.
 - (b) Que se reconsidera la decisión ya tomada anteriormente respecto a la selección de un elemento a la vista de la decisión que se debe tomar en el instante actual.
 - (c) Que la solución no sea la óptima.
13. Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?
- (a) Programación dinámica.
 - (b) Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
 - (c) El método voraz
14. Sea la siguiente relación de recurrencia

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

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

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

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

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

```

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(XXXXXXXX) );
    }
    r[n]=q;
    return q;
}

```

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

22. Una de estas tres situaciones no es posible:

- (a) $f(n) \in O(n)$ y $f(n) \in \Omega(1)$
- (b) $f(n) \in \Omega(n^2)$ y $f(n) \in O(n)$
- (c) $f(n) \in O(n)$ y $f(n) \in O(n^2)$

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

- (a) La complejidad temporal de la mejor solución posible al problema es $O(n!)$.
- (b) Si se construye una solución al problema basada en el esquema de ramificación y poda, una buena elección de cotas optimistas y pesimistas podría evitar la exploración de todas las permutaciones posibles.
- (c) La complejidad temporal de la mejor solución posible al problema es $O(n^2)$.

24. Cuál de los siguientes algoritmos proveería una cota pesimista para el problema de encontrar el camino mas corto entre dos ciudades (se supone que el grafo es conexo).
- (a) Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.
 - (b) Para todas las ciudades que son alcanzables en un paso desde la ciudad inicial, sumar la distancia a dicha ciudad y la distancia geométrica hasta la ciudad destino.
 - (c) Calcular la distancia recorrida moviéndose al azar por el grafo hasta llegar (por azar) a la ciudad destino.
25. Si para resolver un mismo problema usamos un algoritmo de vuelta atrás y lo modificamos mínimamente para convertirlo en un algoritmo de ramificación y poda, ¿qué cambiamos realmente?
- (a) Cambiamos la función que damos a la cota pesimista.
 - (b) El algoritmo puede aprovechar mejor las cotas optimistas.
 - (c) La comprobación de las soluciones factibles: en ramificación y poda no es necesario puesto que sólo genera nodos factibles.
26. Dadas las siguientes funciones:
- ```
// Precondición: { 0 <= i < v.size(); i < j <= v.size() }
unsigned f(const vector<unsigned>&v, unsigned i, unsigned j) {
 if(i == j+1)
 return v[i];
 unsigned sum = 0;
 for(unsigned k = 0; k < j - i; k++)
 sum += f(v, i, i+k+1) + f(v, i+k+1, j);
 return sum;
}

unsigned g(const vector<unsigned>&v) {
 return f(v, v.begin(), v.end());
}
```
- Se quiere reducir la complejidad temporal de la función  $g$  usando programación dinámica iterativa. ¿cuál sería la complejidad espacial?
- (a) cúbica
  - (b) cuadrática
  - (c) exponencial
27. En una cuadrícula se quiere dibujar el contorno de un cuadrado de  $n$  casillas de lado. ¿cuál será la complejidad temporal del mejor algoritmo que pueda existir?
- (a)  $O(n^2)$
  - (b)  $O(n)$
  - (c)  $O(\sqrt{n})$

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

33. ¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz (*greedy*) que es óptima?
- (a) El problema de la mochila discreta.
  - (b) El problema de la mochila continua o con fraccionamiento.
  - (c) El árbol de cobertura de coste mínimo de un grafo conexo.
34. En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,
- (a) una estrategia voraz puede ser la única alternativa.
  - (b) es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione.
  - (c) podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.
35. Dado un problema de optimización, el método voraz...
- (a) ... siempre obtiene la solución óptima.
  - (b) ... garantiza la solución óptima sólo para determinados problemas.
  - (c) ... siempre obtiene una solución factible.
36. La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...
- (a) ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.
  - (b) El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.
  - (c) ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos.
37. Se quieren ordenar  $d$  números distintos comprendidos entre 1 y  $n$ . Para ello se usa un array de  $n$  booleanos que se inicializan primero a *false*. A continuación se recorren los  $d$  números cambiando los valores del elemento del vector de booleanos correspondiente a su número a *true*. Por último se recorre el vector de booleanos escribiendo los índices de los elementos del vector de booleanos que son *true*. ¿Es este algoritmo más rápido (asintóticamente) que el *mergesort*?
- (a) Sí, ya que el *mergesort* es  $O(n \log n)$  y este es  $O(n)$
  - (b) Sólo si  $d \log d > k n$  (donde  $k$  es una constante que depende de la implementación)
  - (c) No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.

38. ¿Cuál de estos problemas tiene una solución eficiente utilizando *programación dinámica*?
- (a) El problema de la asignación de tareas.
  - (b) El problema del cambio.
  - (c) La mochila discreta sin restricciones adicionales.
39. Di cuál de estos tres algoritmos no es un algoritmo de “divide y vencerás”
- (a) Quicksort
  - (b) Mergesort
  - (c) El algoritmo de Prim
40. Si  $f(n) \in O(n^3)$ , ¿puede pasar que  $f(n) \in O(n^2)$ ?
- (a) No, porque  $n^3 \notin O(n^2)$
  - (b) Sólo para valores bajos de  $n$
  - (c) Es perfectamente posible, ya que  $O(n^2) \subset O(n^3)$

# Respuestas para la modalidad F

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

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

2. 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  $O(n^2)$ .

3. 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

4. Sea la siguiente relación de recurrencia

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

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

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

5. ¿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.

6. 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.

7. 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)  $O(n^2)$

8. 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.

9. Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?
- (a) Se multiplica  $k$  por la distancia de la arista más corta que nos queda por considerar, donde  $k$  es el número de saltos que nos quedan por dar.
  - (b) Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
  - (c) Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las  $k$  aristas más cortas, donde  $k$  es el número de saltos que nos quedan por dar.
10. 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 recursiva 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.
11. 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.
12. 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  $O(n^2) \subset O(n^3)$
13. 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.

14. 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.
15. 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.
16. 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.

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

```
void fill(price 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(XXXXXXXX));
 }
 r[n]=q;
 return q;
}
```

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

18. 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.

19. 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.

20. 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.
21. 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.
22. 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
23. 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.
24. 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  $f(n) \in O(n^2)$

25. 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.
26. 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.
27. 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.
28. 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.
29. 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.

30. 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)  $O(\sqrt{n})$
31. Se desea encontrar el camino más corto entre dos ciudades. Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo,  $-1$ ) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geométrica (en línea recta) entre cada par de ciudades. Se pretende acelerar la búsqueda de un algoritmo de *ramificación y poda* priorizando los nodos vivos (ciudades) que estén a menor distancia geográfica de la ciudad objetivo.
- (a) El nuevo algoritmo no garantiza que vaya a ser más rápido para todas las instancias del problema posibles.
  - (b) El nuevo algoritmo siempre será más rápido.
  - (c) Esta estrategia no asegura que se obtenga el camino más corto.
32. 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
33. 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.
34. ¿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.

35. ¿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.
36. Cuando se usa un algoritmo voraz para abordar la resolución de un problema de optimización por selección discreta (es decir, un problema para el cual la solución consiste en encontrar un subconjunto del conjunto de elementos que optimiza una determinada función), ¿cuál de estas tres cosas es imposible que ocurra?
- (a) Que el algoritmo no encuentre ninguna solución.
  - (b) Que se reconsidera la decisión ya tomada anteriormente respecto a la selección de un elemento a la vista de la decisión que se debe tomar en el instante actual.
  - (c) Que la solución no sea la óptima.
37. 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.
38. 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.
39. 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.

40. 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.

# Respuestas para la modalidad C

1. 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 recorrida moviéndose al azar por el grafo hasta llegar (por azar) a la ciudad destino.
- (b) Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.
- (c) 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.

2. 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(XXXXXXXX));
 }
 r[n]=q;
 return q;
}
```

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

3. Sea la siguiente relación de recurrencia

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

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

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

4. Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

- (a) Se 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.
- (b) 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.
- (c) Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.

5. Si un problema de optimización lo es para una función que toma valores continuos ...

- (a) La programación dinámica recursiva puede resultar mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.
- (b) La programación dinámica iterativa siempre es mucho más eficiente que la programación dinámica recursiva 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.

6. 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) Sólo si  $d \log d > k n$  (donde  $k$  es una constante que depende de la implementación)
- (b) Sí, ya que el *mergesort* es  $O(n \log n)$  y este es  $O(n)$
- (c) No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.

7. 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)  $O(\sqrt{n})$
8. El uso de funciones de cota en ramificación y poda...
- (a) ... transforma en polinómicas complejidades que antes eran exponenciales.
  - (b) ... puede reducir el número de instancias del problema que pertenecen al caso peor.
  - (c) ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.
9. 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 dos anteriores son ciertas.
10. Garantiza el uso de una estrategia “divide y vencerás” la existencia de una solución de complejidad temporal polinómica a cualquier problema?
- (a) No
  - (b) Sí, en cualquier caso.
  - (c) Sí, pero siempre que la complejidad temporal conjunta de las operaciones de descomposición del problema y la combinación de las soluciones sea polinómica.
11. 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 convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.
  - (b) Se puede mejorar la eficiencia del algoritmo guardando en una tabla el valor devuelto para cada conjunto de parámetros de cada llamada cuando ésta se produce por primera vez.
  - (c) Se puede mejorar la eficiencia del algoritmo definiendo de antemano el orden en el que se deben calcular las soluciones a los subproblemas y llenando una tabla en ese orden.

12. 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 la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos
  - (c) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido
13. En los algoritmos de *ramificación y poda* ...
- (a) Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.
  - (b) Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.
  - (c) Una cota pesimista es el valor que a lo sumo alcanza cualquier nodo factible que no es el óptimo.
14. 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) ... suele ser polinómica con el número de alternativas por cada decisión.
  - (c) ... puede ser polinómica con el número de decisiones a tomar.
15. Una de estas tres situaciones no es posible:
- (a)  $f(n) \in \Omega(n^2)$  y  $f(n) \in O(n)$
  - (b)  $f(n) \in O(n)$  y  $f(n) \in \Omega(1)$
  - (c)  $f(n) \in O(n)$  y  $f(n) \in O(n^2)$
16. 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) El algoritmo puede aprovechar mejor las cotas optimistas.
  - (b) Cambiamos la función que damos a la cota pesimista.
  - (c) La comprobación de las soluciones factibles: en ramificación y poda no es necesario puesto que sólo genera nodos factibles.
17. Un algoritmo recursivo basado en el esquema *divide y vencerás* ...
- (a) ... será más eficiente cuanto más equitativa sea la división en subproblemas.
  - (b) ... nunca tendrá una complejidad exponencial.
  - (c) Las dos anteriores son verdaderas.

18. Dado un problema de optimización, el método voraz...
- (a) ... siempre obtiene la solución óptima.
  - (b) ... siempre obtiene una solución factible.
  - (c) ... garantiza la solución óptima sólo para determinados problemas.
19. 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 muchas veces un número relativamente pequeño de subproblemas distintos.
  - (b) ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.
  - (c) El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.
20. 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 peor cuando el vector ya está ordenado.
  - (c) ... se comporta mejor cuando el vector ya está ordenado.
21. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...
- (a) ... pueden eliminar soluciones parciales que son factibles.
  - (b) ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
  - (c) Las dos anteriores son verdaderas.
22. 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 inferior para el valor óptimo que a veces puede ser igual a este.
  - (c) ... una cota superior para el valor óptimo.

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

24. 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) cuadrática
- (b) cúbica
- (c) exponencial

25. ¿Para cuál de estos problemas de optimización existe una solución voraz?

- (a) El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
- (b) El problema de la mochila discreta.
- (c) El problema de la asignación de coste mínimo de  $n$  tareas a  $n$  trabajadores cuando el coste de asignar la tarea  $i$  al trabajador  $j$ ,  $c_{ij}$  está tabulado en una matriz.

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

27. ¿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.
28. 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)  $O(n^2)$
  - (c)  $O(n^3)$
29. Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?
- (a) Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
  - (b) Programación dinámica.
  - (c) El método voraz
30. 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, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.
  - (b) Sí, tanto si se usan cotas optimistas para podar el árbol de búsqueda como si no.
  - (c) No, ya que en cualquier caso se deben explorar todas las soluciones factibles.
31. Si  $f(n) \in O(n^3)$ , ¿puede pasar que  $f(n) \in O(n^2)$ ?
- (a) Es perfectamente posible, ya que  $O(n^2) \subset O(n^3)$
  - (b) Sólo para valores bajos de  $n$
  - (c) No, porque  $n^3 \notin O(n^2)$

32. 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  $O(n^2)$ .
  - (b) La complejidad temporal de la mejor solución posible al problema es  $O(n!)$ .
  - (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.
33. En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,
- (a) es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione.
  - (b) podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.
  - (c) una estrategia voraz puede ser la única alternativa.
34. La complejidad temporal en el mejor de los casos...
- (a) ... es una función del tamaño o talla del problema que tiene que estar definida para todos los posibles valores de ésta.
  - (b) ... es el tiempo que tarda el algoritmo en resolver el problema de tamaño o talla más pequeña que se le puede presentar.
  - (c) Las dos anteriores son verdaderas.
35. Uno de estos tres problemas no tiene una solución eficiente que siga el esquema de programación dinámica
- (a) El problema de las torres de Hanoi
  - (b) El problema de la mochila discreta.
  - (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.
36. Di cuál de estos tres algoritmos no es un algoritmo de “divide y vencerás”
- (a) El algoritmo de Prim
  - (b) Quicksort
  - (c) Mergesort

37. 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 maximización, aunque en ocasiones ambos valores pueden coincidir.
  - (c) En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
38. 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 siempre sera más rápido.
  - (b) El nuevo algoritmo no garantiza que vaya a ser más rápido para todas las instancias del problema posibles.
  - (c) Esta estrategia no asegura que se obtenga el camino mas corto.
39. 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) Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.
  - (c) 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.
40. La mejor solución que se conoce para el problema de la mochila continua sigue el esquema ...
- (a) ... voraz.
  - (b) ... divide y vencerás.
  - (c) ... ramificación y poda.

# Respostes per a la modalitat 1

1. Siga la relació de recurrència següent:

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

Si  $T(n) \in O(n)$ , en quin d'aquests tres casos ens podem trobar?

- (a)  $g(n) = \log n$
- (b)  $g(n) = 1$
- (c)  $g(n) = n$

2. Quin d'aquests problemes té una solució eficient utilitzant *programació dinàmica*?

- (a) El problema de l'assignació de tasques.
  - (b) La motilla discreta sense restriccions addicionals.
  - (c) El problema del canvi (retornar una quantitat de diners amb el mínim nombre de monedes).
3. En els algorismes de *backtracking*, pot el valor d'una fita pessimista ser més gran que el valor d'una fita optimista? (s'entén que ambdues fites s'apliquen sobre el mateix node)

- (a) No, el valor de la fita pessimista d'un node mai pot ser superior al de la fita optimista d'aquest mateix node.
- (b) En general sí, si es tracta d'un problema de maximització, encara que en ocasions els dos valors poden coincidir.
- (c) En general sí, si es tracta d'un problema de minimització, encara que en ocasions els dos valors poden coincidir.

4. Per a què pot servir la fita pessimista d'un node de *ramificació i poda*?

- (a) Per actualitzar el valor de la millor solució fins al moment.
- (b) Per descartar el node si no és prometedor.
- (c) Per obtenir una fita optimista més precisa.

5. De les expressions següents, o bé dues són certes i una falsa, o bé dues són falses i una certa. Marca la que (en aquest sentit) és diferent a les altres dues.

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

6. Donada la relació de recurrència:

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

(on  $p$  i  $a$  són enters majors que 1 i  $g(n) = n^k$ ), què ha d'ocórrer perquè es complisca  $T(n) \in \Theta(n^k)$ ?

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

7. La relació de recurrència següent expressa la complexitat d'un algorisme recursiu, on  $g(n)$  és una funció polinòmica:

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

Digues quina de les següents afirmacions és certa:

- (a) Si  $g(n) \in \Theta(n)$  la relació de recurrència representa la complexitat temporal en el cas pitllor de l'algorisme de ordenació *quicksort*.
- (b) Si  $g(n) \in \Theta(n)$  la relació de recurrència representa la complexitat temporal en el cas millor de l'algorisme de ordenació *quicksort*.
- (c) Si  $g(n) \in \Theta(1)$  la relació de recurrència representa la complexitat temporal en el cas millor de l'algorisme d'ordenació *mergesort*.

8. Quina és la complexitat temporal de la següent funció?

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

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

9. Quina és la complexitat temporal en el millor dels casos de la següent funció?

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

- (a)  $\Omega(1)$
- (b)  $\Omega(n)$
- (c) Aquesta funció no té cas millor.

10. Davant un problema d'optimització resolt mitjançant *backtracking*, pot ocórrer que l'ús de les fites pessimistes i optimistes siga inútil, o fins i tot perjudicial?

- (a) Sí, ja que és possible que a pesar d'utilitzar aquestes fites no es descarte cap node.
- (b) Segons el tipus de fita; les pessimistes pot ser que no descarten cap node però l'ús de fites optimistes garanteix la reducció l'espai de recerca.
- (c) No, les fites tant optimistes com a pessimistes garanteixen la reducció de l'espai de solucions i per tant l'eficiència de l'algorisme.

11. Indiqueu quina d'aquestes afirmacions és *certa*.

- (a) La memoïtzació evita que un algorisme recursiu ingenu resolga repetidament el mateix problema.
- (b) L'avantatge de la solució de programació dinàmica iterativa al problema de la motxilla discreta és que mai es realitzen càlculs innecessaris.
- (c) Els algorismes iteratius de programació dinàmica utilitzen memoïtzació per evitar resoldre de nou els mateixos subproblemes que es tornen a presentar.

12. Per a quin d'aquests problemes d'optimització es coneix almenys una solució voraç?
- (a) L'arbre de recobriment mínim per a un graf no dirigit amb pesos.
- (b) El problema de la motxilla discreta.
- (c) El problema de l'assignació de cost mínim de  $n$  tasques a  $n$  treballadors quan el cost d'assignar la tasca  $i$  al treballador  $j$ ,  $c_{ij}$  està tabulat en una matriu.
13. Garanteix l'ús d'una estratègia "divideix i venceràs" l'existència d'una solució de complexitat temporal polinòmica a qualsevol problema?
- (a) No
- (b) Sí, en qualsevol cas.
- (c) Sí, però sempre que la complexitat temporal conjunta de les operacions de descomposició del problema i la combinació de les solucions siga polinòmica.
14. La complexitat temporal de la solució via tornada arrere al problema de la motxilla discreta sense fraccionament és...
- (a) ... exponencial en el cas pitjor.
- (b) ... quadràtica en el cas pitjor.
- (c) ... exponencial en qualsevol cas.
15. Quin dels criteris següents proporcionaria una fita optimista per al problema de trobar el camí mes curt entre dues ciutats (se suposa que el graf és connex)?.
- (a) Calcular la distància recorreguda movent-se a l'atzar pel graf fins a arribar (per atzar) a la ciutat de destinació.
- (b) Calcular la distància geomètrica (en línia recta) entre les ciutats d'origen i de destinació.
- (c) Utilitzar la solució (subóptima) que s'obté quan es resol el problema mitjançant un algorisme voraç.
16. Es vol reduir la complexitat temporal de la següent funció fent ús de programació dinàmica. Quin seria la complexitat temporal resultant?

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

- (a) Es pot reduir fins a lineal.
- (b) Cuadràtica
- (c) La funció no compleix amb els requisits necessaris per a poder aplicar programació dinàmica.

17. Si  $\lim_{n \rightarrow \infty} (f(n)/n^2) = k$ , i  $k \neq 0$ , quina d'aquestes tres afirmacions és certa?
- (a)  $f(n) \in \Theta(n^2)$
  - (b)  $f(n) \in \Omega(n^3)$
  - (c)  $f(n) \in \Theta(n^3)$
18. Siga  $A$  una matriu quadrada  $n \times n$ . Es tracta de buscar una permutació de les columnes tal que la suma dels elements de la diagonal de la matriu resultant siga mínima. Indiqueu quina de les següents afirmacions és certa.
- (a) La complexitat temporal de la millor solució possible al problema és  $O(n \log n)$ .
  - (b) La complexitat temporal de la millor solució possible al problema està en  $\Omega(n^n)$ .
  - (c) Si es construeix una solució al problema basada en l'esquema de ramificació i poda, una bona elecció de fites optimistes i pessimistes podria evitar l'exploració de totes les permutacions possibles.
19. De les expressions següents, o bé dues són certes i una falsa, o bé dues són falses i una certa. Marca la que (en aquest sentit) és diferent a les altres dues.
- (a)  $\Theta(f) = O(f) \cap \Omega(f)$
  - (b)  $O(f) = \Omega(f) \cap \Theta(f)$
  - (c)  $\Omega(f) = \Theta(f) \cap O(f)$
20. De les expressions següents, o bé dues són certes i una falsa, o bé dues són falses i una certa. Marca la que (en aquest sentit) és diferent a les altres dues.
- (a)  $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$
  - (b)  $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$
  - (c)  $(4^{\log_2(n)}) \subset O(n) \subset O(2^n)$
21. Quan es resol el problema de la motxilla discreta usant l'estrategia de tornada arrere, pot ocórrer que es tarde menys a trobar la solució óptima si es prova primer a ficar cada objecte abans de no ficar-ho?
- (a) Sí, però només si s'usen fites optimistes per podar l'arbre de recerca.
  - (b) Sí, tant si s'usen fites optimistes per podar l'arbre de recerca com si no.
  - (c) No, ja que en qualsevol cas s'han d'explorar totes les solucions factibles.

22. En els algorismes de *ramificació i poda* ...

- (a) Una fita optimista és necessàriament un valor insuperable; si no fóra així es podria podar el node que condueix a la solució òptima.
- (b) Una fita optimista és necessàriament un valor assolible; si no és així no està garantit que es trobe la solució òptima.
- (c) Una fita pessimista és el valor que com a màxim aconsegueix qualsevol node factible que no és l'òptim.

23. En un algorisme de *ramificació i poda*, l'ordre escollit per prioritzar els nodes en la llista de nodes vius ...

- (a) ... mai afecta al temps necessari per trobar la solució òptima.
- (b) ... determina la complexitat temporal en el pitjor dels casos de l'algorisme.
- (c) ... pot influir en el nombre de nodes que es descarten sense arribar a expandir-los.

24. Els algorismes de *tornada arrere* que fan ús de fites optimistes generen les solucions possibles al problema mitjançant ...

- (a) ... un recorregut guiat per les que poden ser les millors branques de l'arbre que representa l'espai de solucions.
- (b) ... un recorregut en profunditat de l'arbre que representa l'espai de solucions.
- (c) ... un recorregut guiat per una cua de prioritat d'on s'extrauen primer els nodes que representen els subarbres més prometedors de l'espai de solucions.

25. Què s'entén per *grandària del problema*?

- (a) La quantitat d'espai en memòria que es necessita per codificar una instància d'aquest problema.
- (b) El valor màxim que pot prendre una instància qualsevol d'aquest problema.
- (c) El nombre de paràmetres que componen el problema.

26. L'algorisme d'ordenació *Quicksort* divideix el problema en dos subproblems. Quina és la complexitat temporal asimptòtica de realitzar aquesta divisió?

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

27. Es vol ordenar  $d$  números diferents compresos entre 1 i  $n$ . Per a això s'usa un array de  $n$  booleans que s'inicialitzen primer a *false*. A continuació es recorren els  $d$  números canviant els valors de l'element del vector de booleans corresponent al seu números a *true*. Finalment es recorre el vector de booleans escrivint els índexs dels elements del vector de booleans que són *true*. És aquest algorisme més ràpid (asimptòticament) que el *mergesort*?
- (a) Només si  $d \log d > kn$  (on  $k$  és una constant que depén de la implementació)
  - (b) Sí, ja que el *mergesort* és  $O(n \log n)$  i aquest és  $O(n)$
  - (c) No, ja que aquest algorisme ha de recórrer diverses vegades el vector de booleans.
28. En el problema de l'acolorit de grafs (mínim nombre de colors necessaris per a acolorir tots els vèrtexs d'un graf de manera que no queden dos adjacents amb el mateix color) resolt mitjançant *ramificació i poda*, de quina manera s'hauria d'ordenar la llista de nodes vius per a obtenir una solució acceptable?
- (a) Per cota optimista: explorant primer els nodes amb menor cota optimista.
  - (b) Per cota pessimista: explorant primer els nodes amb menor cota optimista.
  - (c) les altres dues opcions poden ser ambdues correctes.
29. Quina és la definició correcta de  $\Omega(g)$ ?
- (a)  $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ | \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \geq cg(n)\}$
  - (b)  $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ | \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$
  - (c)  $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ | \forall c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$
30. La solució recursiva ingènua (però correcta) a un problema d'optimització crida més d'una vegada a la funció amb els mateixos paràmetres. Una de les tres afirmacions següents és falsa.
- (a) Es pot millorar l'eficiència de l'algorisme convertint l'algorisme recursiu directament en iteratiu sense canviar el seu funcionament bàsic.
  - (b) Es pot millorar l'eficiència de l'algorisme guardant en una taula el valor retornat per a cada conjunt de paràmetres de cada crida quan aquesta es produeix per primera vegada.
  - (c) Es pot millorar l'eficiència de l'algorisme definint per endavant l'ordre en el qual s'han de calcular les solucions als subproblemes i omplint una taula en aquest ordre.

31. En l'esquema de *tornada enrere*, els mecanismes de poda basats en la millor solució fins al moment...
- (a) ... poden eliminar solucions parcials que són factibles.
  - (b) ... garanteixen que no s'explorarà mai tot l'espai de solucions possibles.
  - (c) Les altres dues opcions són ambdues certes.
32. En el problema de l'acolorit de grafs (mínim nombre de colors necessaris per a acolorir tots els vèrtexs d'un graf de manera que no queden dos adjacents amb el mateix color) resolt mitjançant *ramificació i poda*, una cota optimista és el resultat de l'assumir que ...
- (a) ... no es van a utilitzar colors diferents als ja utilitzats.
  - (b) ... es van a utilitzar tants colors diferents als ja utilitzats com a vèrtexs queden per acolorir.
  - (c) ... només vaa ser necessari un color més.
33. Considerem l'algorisme d'ordenació *Mergesort* modificat de manera que, en comptes de dividir el vector en dues parts, es divideix en tres. Posteriorment es combinen les solucions parcials. Quina seria la complexitat temporal asimptòtica de la combinació de les solucions parcials?
- (a) Cap de les altres dues opcions és certa.
  - (b)  $\Theta(n)$
  - (c)  $\Theta(\log_3 n)$
34. Es vol reduir la complexitat temporal de la funció  $g$  usant programació dinàmica iterativa. Quina seria la complexitat espacial de l'algorisme resultant?
- ```
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) Quadràtica
 - (b) Lineal
 - (c) Cúbica
35. Si $f \in \Theta(g_1)$ y $f \in \Theta(g_2)$ aleshores
- (a) $f \notin \Theta(\max(g_1, g_2))$
 - (b) $f \in \Theta(g_1 \cdot g_2)$
 - (c) $f \in \Theta(g_1 + g_2)$

Respostes per a la modalitat 2

1. En el problema de l'acolorit de grafs (mínim nombre de colors necessaris per a acolorir tots els vèrtexs d'un graf de manera que no queden dos adjacents amb el mateix color) resolt mitjançant *ramificació i poda*, de quina manera s'hauria d'ordenar la llista de nodes vius per a obtenir una solució acceptable?
 - (a) Per cota optimista: explorant primer els nodes amb menor cota optimista.
 - (b) Per cota pessimista: explorant primer els nodes amb menor cota optimista.
 - (c) les altres dues opcions poden ser ambdues correctes.
2. La complexitat temporal de la solució via tornada arrere al problema de la motxilla discreta sense fraccionament és...
 - (a) ... exponencial en el cas pitjor.
 - (b) ... quadràtica en el cas pitjor.
 - (c) ... exponencial en qualsevol cas.
3. Els algorismes de *tornada arrere* que fan ús de fites optimistes generen les solucions possibles al problema mitjançant ...
 - (a) ... un recorregut guiat per les que poden ser les millors branques de l'arbre que representa l'espai de solucions.
 - (b) ... un recorregut en profunditat de l'arbre que representa l'espai de solucions.
 - (c) ... un recorregut guiat per una cua de prioritat d'on s'extrauen primer els nodes que representen els subarbres més prometedors de l'espai de solucions.
4. Quan es resol el problema de la motxilla discreta usant l'estrategia de tornada arrere, pot ocórrer que es tarde menys a trobar la solució òptima si es prova primer a ficar cada objecte abans de no ficar-ho?
 - (a) Sí, però només si s'usen fites optimistes per podar l'arbre de recerca.
 - (b) Sí, tant si s'usen fites optimistes per podar l'arbre de recerca com si no.
 - (c) No, ja que en qualsevol cas s'han d'explorar totes les solucions factibles.
5. En els algorismes de *ramificació i poda* ...
 - (a) Una fita optimista és necessàriament un valor insuperable; si no fóra així es podria podar el node que condueix a la solució òptima.
 - (b) Una fita optimista és necessàriament un valor assolible; si no és així no està garantit que es trobe la solució òptima.
 - (c) Una fita pessimista és el valor que com a màxim aconsegueix qualsevol node factible que no és l'òptim.

6. Davant un problema d'optimització resolt mitjançant *backtracking*, pot ocurrir que l'ús de les fites pessimistes i optimistes siga inútil, o fins i tot perjudicial?
- (a) Sí, ja que és possible que a pesar d'utilitzar aquestes fites no es descarte cap node.
- (b) Segons el tipus de fita; les pessimistes pot ser que no descarten cap node però l'ús de fites optimistes garanteix la reducció l'espai de recerca.
- (c) No, les fites tant optimistes com a pessimistes garanteixen la reducció de l'espai de solucions i per tant l'eficiència de l'algorisme.
7. Si $f \in \Theta(g_1)$ y $f \in \Theta(g_2)$ aleshores
- (a) $f \notin \Theta(\max(g_1, g_2))$
- (b) $f \in \Theta(g_1 \cdot g_2)$
- (c) $f \in \Theta(g_1 + g_2)$
8. En un algorisme de *ramificació i poda*, l'ordre escollit per prioritzar els nodes en la llista de nodes vius ...
- (a) ... mai afecta al temps necessari per trobar la solució óptima.
- (b) ... determina la complexitat temporal en el pitjor dels casos de l'algorisme.
- (c) ... pot influir en el nombre de nodes que es descarten sense arribar a expandir-los.
9. Si $\lim_{n \rightarrow \infty} (f(n)/n^2) = k$, i $k \neq 0$, quina d'aquestes tres afirmacions és *certa*?
- (a) $f(n) \in \Theta(n^2)$
- (b) $f(n) \in \Omega(n^3)$
- (c) $f(n) \in \Theta(n^3)$
10. Indiqueu quina d'aquestes afirmacions és *certa*.
- (a) La memoïtzació evita que un algorisme recursiu ingenu resolga repetidament el mateix problema.
- (b) L'avantatge de la solució de programació dinàmica iterativa al problema de la motxilla discreta és que mai es realitzen càlculs innecessaris.
- (c) Els algorismes iteratius de programació dinàmica utilitzen memoïtzació per evitar resoldre de nou els mateixos subproblemes que es tornen a presentar.

11. La relació de recurrència següent expressa la complexitat d'un algorisme recursiu, on $g(n)$ és una funció polinòmica:

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

Diges quina de les següents afirmacions és certa:

- (a) Si $g(n) \in \Theta(n)$ la relació de recurrència representa la complexitat temporal en el cas pitllor de l'algorisme de ordenació *quicksort*.
- (b) Si $g(n) \in \Theta(n)$ la relació de recurrència representa la complexitat temporal en el cas millor de l'algorisme de ordenació *quicksort*.
- (c) Si $g(n) \in \Theta(1)$ la relació de recurrència representa la complexitat temporal en el cas millor de l'algorisme d'ordenació *mergesort*.

12. Siga la relació de recurrència següent:

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

Si $T(n) \in O(n)$, en quin d'aquests tres casos ens podem trobar?

- (a) $g(n) = \log n$
- (b) $g(n) = 1$
- (c) $g(n) = n$

13. Què s'entén per *grandària del problema*?

- (a) La quantitat d'espai en memòria que es necessita per codificar una instància d'aquest problema.
- (b) El valor màxim que pot prendre una instància qualsevol d'aquest problema.
- (c) El nombre de paràmetres que componen el problema.

14. Es vol reduir la complexitat temporal de la funció g usant programació dinàmica iterativa. Quina seria la complexitat espacial de l'algorisme resultant?

```
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) Quadràtica
- (b) Lineal
- (c) Cúbica

15. Quin dels criteris següents proporcionaria una fita optimista per al problema de trobar el camí més curt entre dues ciutats (se suposa que el graf és connex)?.

- (a) Calcular la distància recorreguda movent-se a l'atzar pel graf fins a arribar (per atzar) a la ciutat de destinació.
- (b) Calcular la distància geomètrica (en línia recta) entre les ciutats d'origen i de destinació.
- (c) Utilitzar la solució (subóptima) que s'obté quan es resol el problema mitjançant un algorisme voraç.

16. Quina és la complexitat temporal en el millor dels casos de la següent funció?

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

- (a) $\Omega(1)$
- (b) $\Omega(n)$
- (c) Aquesta funció no té cas millor.

17. Donada la relació de recurrència:

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

(on p i a són enters majors que 1 i $g(n) = n^k$), què ha d'ocórrer perquè es complisca $T(n) \in \Theta(n^k)$?

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

18. La solució recursiva ingènua (però correcta) a un problema d'optimització crida més d'una vegada a la funció amb els mateixos paràmetres. Una de les tres afirmacions següents és falsa.
- (a) Es pot millorar l'eficiència de l'algorisme convertint l'algorisme recursiu directament en iteratiu sense canviar el seu funcionament bàsic.
 - (b) Es pot millorar l'eficiència de l'algorisme guardant en una taula el valor retornat per a cada conjunt de paràmetres de cada crida quan aquesta es produeix per primera vegada.
 - (c) Es pot millorar l'eficiència de l'algorisme definint per endavant l'ordre en el qual s'han de calcular les solucions als subproblemes i omplint una taula en aquest ordre.
19. De les expressions següents, o bé dues són certes i una falsa, o bé dues són falses i una certa. Marca la que (en aquest sentit) és diferent a les altres dues.
- (a) $\Theta(\log^2(n)) = \Theta(\log^3(n))$
 - (b) $\log(n^3) \notin \Theta(\log_3(n))$
 - (c) $\Theta(\log_2(n)) = \Theta(\log_3(n))$
20. L'algorisme d'ordenació *Quicksort* divideix el problema en dos subproblems. Quina és la complexitat temporal asimptòtica de realitzar aquesta divisió?
- (a) $O(1)$
 - (b) $O(n)$
 - (c) $O(n \log n)$
21. En el problema de l'acolorit de grafs (mínim nombre de colors necessaris per a acolorir tots els vèrtexs d'un graf de manera que no queden dos adjacents amb el mateix color) resolt mitjançant *ramificació i poda*, una cota optimista és el resultat de l'assumir que ...
- (a) ...no es van a utilitzar colors diferents als ja utilitzats.
 - (b) ...es van a utilitzar tants colors diferents als ja utilitzats com a vèrtexs queden per acolorir.
 - (c) ...només vaa ser necessari un color més.
22. Garanteix l'ús d'una estratègia "divideix i venceràs" l'existència d'una solució de complexitat temporal polinòmica a qualsevol problema?
- (a) No
 - (b) Sí, en qualsevol cas.
 - (c) Sí, però sempre que la complexitat temporal conjunta de les operacions de descomposició del problema i la combinació de les solucions siga polinòmica.

23. Quina és la definició correcta de $\Omega(g)$?

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

24. De les expressions següents, o bé dues són certes i una falsa, o bé dues són falses i una certa. Marca la que (en aquest sentit) és diferent a les altres dues.

- (a) $\Theta(f) = O(f) \cap \Omega(f)$
(b) $O(f) = \Omega(f) \cap \Theta(f)$
(c) $\Omega(f) = \Theta(f) \cap O(f)$

25. En l'esquema de *tornada enrere*, els mecanismes de poda basats en la millor solució fins al moment...

- (a) ... poden eliminar solucions parcials que són factibles.
(b) ... garanteixen que no s'explorarà mai tot l'espai de solucions possibles.
(c) Les altres dues opcions són ambdues certes.

26. Quina és la complexitat temporal de la següent funció?

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

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

27. De les expressions següents, o bé dues són certes i una falsa, o bé dues són falses i una certa. Marca la que (en aquest sentit) és diferent a les altres dues.

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

28. Per a què pot servir la fita pessimista d'un node de *ramificació i poda*?
- (a) Per actualitzar el valor de la millor solució fins al moment.
 - (b) Per descartar el node si no és prometedor.
 - (c) Per obtenir una fita optimista més precisa.
29. Quin d'aquests problemes té una solució eficient utilitzant *programació dinàmica*?
- (a) El problema de l'assignació de tasques.
 - (b) La motxilla discreta sense restriccions addicionals.
 - (c) El problema del canvi (retornar una quantitat de diners amb el mínim nombre de monedes).
30. En els algorismes de *backtracking*, pot el valor d'una fita pessimista ser més gran que el valor d'una fita optimista? (s'entén que ambdues fites s'apliquen sobre el mateix node)
- (a) No, el valor de la fita pessimista d'un node mai pot ser superior al de la fita optimista d'aquest mateix node.
 - (b) En general sí, si es tracta d'un problema de maximització, encara que en ocasions els dos valors poden coincidir.
 - (c) En general sí, si es tracta d'un problema de minimització, encara que en ocasions els dos valors poden coincidir.
31. Es vol ordenar d números diferents compresos entre 1 i n . Per a això s'usa un array de n booleans que s'inicialitzen primer a *false*. A continuació es recorren els d números canviant els valors de l'element del vector de booleans corresponent al seu números a *true*. Finalment es recorre el vector de booleans escrivint els índexs dels elements del vector de booleans que són *true*. És aquest algorisme més ràpid (asimptòticament) que el *mergesort*?
- (a) Només si $d \log d > k n$ (on k és una constant que depén de la implementació)
 - (b) Sí, ja que el *mergesort* és $O(n \log n)$ i aquest és $O(n)$
 - (c) No, ja que aquest algorisme ha de recórrer diverses vegades el vector de booleans.
32. Siga A una matriu quadrada $n \times n$. Es tracta de buscar una permutació de les columnes tal que la suma dels elements de la diagonal de la matriu resultant siga mínima. Indiqueu quina de les següents afirmacions és certa.
- (a) La complexitat temporal de la millor solució possible al problema és $O(n \log n)$.
 - (b) La complexitat temporal de la millor solució possible al problema està en $\Omega(n^n)$.
 - (c) Si es construeix una solució al problema basada en l'esquema de ramificació i poda, una bona elecció de fites optimistes i pessimistes podria evitar l'exploració de totes les permutacions possibles.

33. Es vol reduir la complexitat temporal de la següent funció fent ús de programació dinàmica. Quin seria la complexitat temporal resultant?

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

- (a) Es pot reduir fins a lineal.
 (b) Cuadrática
(c) La funció no compleix amb els requisits necessaris per a poder aplicar programació dinàmica.
34. Considerem l'algorisme d'ordenació *Mergesort* modificat de manera que, en comptes de dividir el vector en dues parts, es divideix en tres. Posteriorment es combinen les solucions parcials. Quina seria la complexitat temporal asymptòtica de la combinació de les solucions parcials?

- (a) Cap de les altres dues opcions és certa.
 (b) $\Theta(n)$
(c) $\Theta(\log_3 n)$
35. Per a quin d'aquests problemes d'optimització es coneix almenys una solució voraç?

- (a) L'arbre de recobriment mínim per a un graf no dirigit amb pesos.
(b) El problema de la motxilla discreta.
(c) El problema de l'assignació de cost mínim de n tasques a n treballadors quan el cost d'assignar la tasca i al treballador j , c_{ij} està tabulat en una matriu.

Respuestas para la modalidad 3

1. Sea la siguiente relación de recurrencia

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

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

- (a) $g(n) = \log n$
- (b) $g(n) = 1$
- (c) $g(n) = n$

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

- (a) El problema de la asignación de tareas.
 - (b) La mochila discreta sin restricciones adicionales.
 - (c) El problema del cambio (devolver una cantidad de dinero con el mínimo número de monedas).
3. En los algoritmos de *backtracking*, ¿Puede el valor de una cota pesimista ser mayor que el valor de una cota optimista? (se entiende que ambas cotas se aplican sobre el mismo nodo)
- (a) No, el valor de la cota pesimista de un nodo nunca puede ser superior al de la cota optimista de ese mismo nodo.
 - (b) En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
 - (c) En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.

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

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

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

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

6. Dada la relación de recurrencia:

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

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

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

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

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

Di cuál de las siguientes afirmaciones es *cierta*:

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

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

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

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

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

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

- (a) $\Omega(1)$
- (b) $\Omega(n)$
- (c) Esta función no tiene caso mejor.

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

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

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

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

12. ¿Para cuál de estos problemas de optimización se conoce al menos una solución voraz óptima?
- (a) El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
 - (b) El problema de la mochila discreta.
 - (c) El problema de la asignación de coste mínimo de n tareas a n trabajadores cuando el coste de asignar la tarea i al trabajador j , c_{ij} está tabulado en una matriz.
13. Garantiza el uso de una estrategia “divide y vencerás” la existencia de una solución de complejidad temporal polinómica a cualquier problema?
- (a) No.
 - (b) Sí, en cualquier caso.
 - (c) Sí, pero siempre que la complejidad temporal conjunta de las operaciones de descomposición del problema y la combinación de las soluciones sea polinómica.
14. La complejidad temporal de la solución de *vuelta atrás* al problema de la mochila discreta es ...
- (a) ...exponencial en el caso peor..
 - (b) ...cuadrática en el caso peor.
 - (c) ...exponencial en cualquier caso.
15. Cuál de los siguientes criterios proporcionaría una cota optimista para el problema de encontrar el camino mas corto entre dos ciudades (se supone que el grafo es conexo).
- (a) Calcular la distancia recorrida moviéndose al azar por el grafo hasta llegar (por azar) a la ciudad destino.
 - (b) Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.
 - (c) Utilizar la solución (subóptima) que se obtiene al resolver el problema mediante un algoritmo voraz.
16. Se quiere reducir la complejidad temporal de la siguiente función haciendo uso de programación dinámica. ¿Cuál sería la complejidad temporal resultante?
- ```
unsigned g(unsigned n, unsigned r){
 if (r==0 || r==n)
 return 1;
 return g(n-1, r-1) + g(n-1, r);
}
```
- (a) Se puede reducir hasta lineal.
  - (b) Cuadrática
  - (c) La función no cumple con los requisitos necesarios para poder aplicar programación dinámica.

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

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

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

- (a) La complejidad temporal de la mejor solución posible al problema es  $O(n \log n)$ .
- (b) La complejidad temporal de la mejor solución posible al problema está en  $\Omega(n^n)$ .
- (c) Si se construye una solución al problema basada en el esquema de ramificación y poda, una buena elección de cotas optimistas y pesimistas podría evitar la exploración de todas las permutaciones posibles.

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

- (a)  $\Theta(f) = O(f) \cap \Omega(f)$
- (b)  $O(f) = \Omega(f) \cap \Theta(f)$
- (c)  $\Omega(f) = \Theta(f) \cap O(f)$

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

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

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

- (a) Sí, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.
- (b) Sí, tanto si se usan cotas optimistas para podar el árbol de búsqueda como si no.
- (c) No, ya que en cualquier caso se deben explorar todas las soluciones factibles.

22. En los algoritmos de *ramificación y poda* ...

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

23. En un algoritmo de *ramificación y poda*, el orden escogido para priorizar los nodos en la lista de nodos vivos ...

- (a) ... nunca afecta al tiempo necesario para encontrar la solución óptima.
- (b) ... determina la complejidad temporal en el peor de los casos del algoritmo.
- (c) ... puede influir en el número de nodos que se descartan sin llegar a expandirlos.

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

- (a) ... un recorrido guiado por estimaciones de las que pueden ser las mejores ramas del árbol que representa el espacio de soluciones.
- (b) ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- (c) ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.

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

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

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

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

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

- (a) Sólo si  $d \log d > k n$  (donde  $k$  es una constante que depende de la implementación)
- (b) Sí, ya que el *mergesort* es  $O(n \log n)$  y este es  $O(n)$
- (c) No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.

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

- (a) Por cota optimista: explorando primero los nodos con menor cota optimista.
- (b) Por cota pesimista: explorando primero los nodos con menor cota pesimista.
- (c) las otras dos opciones pueden ser ambas correctas.

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

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

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

- (a) Se puede mejorar la eficiencia del algoritmo convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.
- (b) Se puede mejorar la eficiencia del algoritmo guardando en una tabla el valor devuelto para cada conjunto de parámetros de cada llamada cuando ésta se produce por primera vez.
- (c) Se puede mejorar la eficiencia del algoritmo definiendo de antemano el orden en el que se deben calcular las soluciones a los subproblemas y llenando una tabla en ese orden.

31. En el esquema de *vuelta atrás*, los mecanismos de poda basados en la mejor solución hasta el momento...

- (a) ... pueden eliminar soluciones parciales que son factibles.
- (b) ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- (c) Las otras dos opciones son ambas son verdaderas.

32. En el problema del coloreado de grafos (mínimo número de colores necesarios para colorear todos los vértices de un grafo de manera que no queden dos adyacentes con el mismo color) resuelto mediante *ramificación y poda*, una cota optimista es el resultado de asumir que ...

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

33. Supongamos el algoritmo de ordenación *Mergesort* modificado de manera que, en lugar de dividir el vector en dos partes, se divide en tres. Posteriormente se combinan las soluciones parciales. ¿Cuál sería la complejidad temporal asintótica de la combinación de las soluciones parciales?

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

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

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

- (a) Cuadrática.
- (b) Lineal.
- (c) Cúbica.

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

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

## Respuestas para la modalidad 4

1. 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) Sí, puesto que es posible que a pesar de utilizar dichas cotas no se descarte ningún nodo.
  - (c) No, las cotas tanto optimistas como pesimistas garantizan la reducción del espacio de soluciones y por tanto la eficiencia del algoritmo.
2. La complejidad temporal de la solución de *vuelta atrás* al problema de la mochila discreta es ...
  - (a) ...cuadrática en el caso peor.
  - (b) ...exponencial en el caso peor..
  - (c) ...exponencial en cualquier caso.
3. ¿Cuál es la definición correcta de  $\Omega(g)$ ?
  - (a)  $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ | \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$
  - (b)  $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ | \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \geq cg(n)\}$
  - (c)  $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ | \forall c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$
4. 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) ... pueden eliminar soluciones parciales que son factibles.
  - (c) Las otras dos opciones son ambas son verdaderas.

5. ¿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)  $\Theta(n)$

6. ¿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.

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

- (a) las otras dos opciones pueden ser ambas correctas.
- (b) Por cota optimista: explorando primero los nodos con menor cota optimista.
- (c) Por cota pesimista: explorando primero los nodos con menor cota pesimista.

8. ¿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.

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

10. Dada la relación de recurrencia:

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

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

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

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

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

12. Sea la siguiente relación de recurrencia

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

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

- (a)  $g(n) = 1$
- (b)  $g(n) = \log n$
- (c)  $g(n) = n$

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

- (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.

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

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

15. 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.

16. 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) Lineal.
- (b) Cuadrática.
- (c) Cúbica.

17. En un algoritmo de *ramificación y poda*, el orden escogido para priorizar los nodos en la lista de nodos vivos ...

- (a) ... determina la complejidad temporal en el peor de los casos del algoritmo.
- (b) ... nunca afecta al tiempo necesario para encontrar la solución óptima.
- (c) ... puede influir en el número de nodos que se descartan sin llegar a expandirlos.

18. 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.

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

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

20. ¿Para cuál de estos problemas de optimización se conoce al menos una solución voraz óptima?
- (a) El problema de la mochila discreta.
  - (b) El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
  - (c) El problema de la asignación de coste mínimo de  $n$  tareas a  $n$  trabajadores cuando el coste de asignar la tarea  $i$  al trabajador  $j$ ,  $c_{ij}$  está tabulado en una matriz.
21. 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)  $\Theta(\log_2(n)) = \Theta(\log_3(n))$
22. Cuál de los siguientes criterios proporcionaría una cota optimista para el problema de encontrar el camino mas corto entre dos ciudades (se supone que el grafo es conexo).
- (a) Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.
  - (b) Calcular la distancia recorrida moviéndose al azar por el grafo hasta llegar (por azar) a la ciudad destino.
  - (c) Utilizar la solución (subóptima) que se obtiene al resolver el problema mediante un algoritmo voraz.
23. 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) Sí, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.
  - (c) No, ya que en cualquier caso se deben explorar todas las soluciones factibles.
24. Supongamos el algoritmo de ordenación *Mergesort* modificado de manera que, en lugar de dividir el vector en dos partes, se divide en tres. Posteriormente se combinan las soluciones parciales. ¿Cuál sería la complejidad temporal asintótica de la combinación de las soluciones parciales?
- (a)  $\Theta(n)$
  - (b)  $\Theta(\log_3 n)$
  - (c) Ninguna de las otras dos opciones es cierta.

25. Los algoritmos de *vuelta atrás* que hacen uso de cotas optimistas generan las soluciones posibles al problema mediante ...
- (a) ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
  - (b) ... un recorrido guiado por estimaciones de las que pueden ser las mejores ramas del árbol que representa el espacio de soluciones.
  - (c) ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.
26. 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.
27. Si  $f \in \Theta(g_1)$  y  $f \in \Theta(g_2)$  entonces
- (a)  $f \in \Theta(g_1 \cdot g_2)$
  - (b)  $f \notin \Theta(\max(g_1, g_2))$
  - (c)  $f \in \Theta(g_1 + g_2)$
28. ¿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 actualizar el valor de la mejor solución hasta el momento.
  - (c) Para obtener una cota optimista más precisa.

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

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

Di cuál de las siguientes afirmaciones es *cierta*:

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

30. 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.

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

32. 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)  $O(f) = \Omega(f) \cap \Theta(f)$

33. 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 convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.
  - (c) Se puede mejorar la eficiencia del algoritmo definiendo de antemano el orden en el que se deben calcular las soluciones a los subproblemas y llenando una tabla en ese orden.
34. Si  $\lim_{n \rightarrow \infty} (f(n)/n^2) = k$ , y  $k \neq 0$ , ¿cuál de estas tres afirmaciones es cierta?
- (a)  $f(n) \in \Omega(n^3)$
  - (b)  $f(n) \in \Theta(n^2)$
  - (c)  $f(n) \in \Theta(n^3)$
35. 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.

## Respuestas para la modalidad 5

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  $\tilde{O}(n \log n)$ .
- (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.

2. Sea la siguiente relación de recurrencia

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

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

- (a)  $g(n) = 1$
- (b)  $g(n) = \log n$
- (c)  $g(n) = n$

3. 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) Sí, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.
- (c) No, ya que en cualquier caso se deben explorar todas las soluciones factibles.

4. 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) Lineal.
- (b) Cuadrática.
- (c) Cúbica.

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

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

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

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

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.

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

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

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

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.
11. ¿Cuál de estos problemas tiene una solución eficiente utilizando *programación dinámica*?
- (a) La mochila discreta sin restricciones adicionales.
  - (b) El problema de la asignación de tareas.
  - (c) El problema del cambio (devolver una cantidad de dinero con el mínimo número de monedas).
12. De las siguientes expresiones, o bien dos son verdaderas y una es falsa, o bien dos son falsas y una es verdadera. Marca la que (en este sentido) es distinta a las otras dos.
- (a)  $\log(n^3) \notin \Theta(\log_3(n))$
  - (b)  $\Theta(\log^2(n)) = \Theta(\log^3(n))$
  - (c)  $\Theta(\log_2(n)) = \Theta(\log_3(n))$
13. Se quiere reducir la complejidad temporal de la siguiente función haciendo uso de programación dinámica. ¿Cuál sería la complejidad temporal resultante?
- ```
unsigned g( unsigned n, unsigned r) {
    if (r==0 || r==n)
        return 1;
    return g(n-1, r-1) + g(n-1, r);
}
```
- (a) Cuadrática
 - (b) Se puede reducir hasta lineal.
 - (c) La función no cumple con los requisitos necesarios para poder aplicar programación dinámica.
14. De las siguientes expresiones, o bien dos son verdaderas y una es falsa, o bien dos son falsas y una es verdadera. Marca la que (en este sentido) es distinta a las otras dos.
- (a) $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$
 - (b) $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$
 - (c) $(4^{\log_2(n)}) \subset O(n) \subset O(2^n)$

15. Dada la relación de recurrencia:

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

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

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

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

- (a) El problema de la mochila discreta.
- (b) El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
- (c) El problema de la asignación de coste mínimo de n tareas a n trabajadores cuando el coste de asignar la tarea i al trabajador j , c_{ij} está tabulado en una matriz.

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

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

18. 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) ... pueden eliminar soluciones parciales que son factibles.
- (c) Las otras dos opciones son ambas son verdaderas.

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

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

20. 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 convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.
 - (c) Se puede mejorar la eficiencia del algoritmo definiendo de antemano el orden en el que se deben calcular las soluciones a los subproblemas y llenando una tabla en ese orden.
21. En el problema del coloreado de grafos (mínimo número de colores necesarios para colorear todos los vértices de un grafo de manera que no queden dos adyacentes con el mismo color) resuelto mediante *ramificación y poda*, ¿De qué manera se debería ordenar la lista de nodos vivos para obtener una solución aceptable?
- (a) las otras dos opciones pueden ser ambas correctas.
 - (b) Por cota optimista: explorando primero los nodos con menor cota optimista.
 - (c) Por cota pesimista: explorando primero los nodos con menor cota pesimista.
22. Si $\lim_{n \rightarrow \infty} (f(n)/n^2) = k$, y $k \neq 0$, ¿cuál de estas tres afirmaciones es cierta?
- (a) $f(n) \in \Omega(n^3)$
 - (b) $f(n) \in \Theta(n^2)$
 - (c) $f(n) \in \Theta(n^3)$
23. 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) Sí, puesto que es posible que a pesar de utilizar dichas cotas no se descarte ningún nodo.
 - (c) No, las cotas tanto optimistas como pesimistas garantizan la reducción del espacio de soluciones y por tanto la eficiencia del algoritmo.
24. Si $f \in \Theta(g_1)$ y $f \in \Theta(g_2)$ entonces
- (a) $f \in \Theta(g_1 \cdot g_2)$
 - (b) $f \notin \Theta(\max(g_1, g_2))$
 - (c) $f \in \Theta(g_1 + g_2)$

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

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

- (a) ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- (b) ... un recorrido guiado por estimaciones de las que pueden ser las mejores ramas del árbol que representa el espacio de soluciones.
- (c) ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.

28. 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.

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

- (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.

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

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

Di cuál de las siguientes afirmaciones es *cierta*:

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

31. En un algoritmo de *ramificación y poda*, el orden escogido para priorizar los nodos en la lista de nodos vivos ...

- (a) ... determina la complejidad temporal en el peor de los casos del algoritmo.
- (b) ... nunca afecta al tiempo necesario para encontrar la solución óptima.
- (c) ... puede influir en el número de nodos que se descartan sin llegar a expandirlos.

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.

33. ¿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 actualizar el valor de la mejor solución hasta el momento.
- (c) Para obtener una cota optimista más precisa.

34. 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.

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.

Respuestas para la modalidad 6

1. En el problema del coloreado de grafos (mínimo número de colores necesarios para colorear todos los vértices de un grafo de manera que no queden dos adyacentes con el mismo color) resuelto mediante *ramificación y poda*, ¿De qué manera se debería ordenar la lista de nodos vivos para obtener una solución aceptable?
 - (a) Por cota optimista: explorando primero los nodos con menor cota optimista.
 - (b) Por cota pesimista: explorando primero los nodos con menor cota pesimista.
 - (c) las otras dos opciones pueden ser ambas correctas.
2. La complejidad temporal de la solución de *vuelta atrás* al problema de la mochila discreta es ...
 - (a) ...exponencial en el caso peor..
 - (b) ...cuadrática en el caso peor.
 - (c) ...exponencial en cualquier caso.
3. Los algoritmos de *vuelta atrás* que hacen uso de cotas optimistas generan las soluciones posibles al problema mediante ...
 - (a) ...un recorrido guiado por estimaciones de las que pueden ser las mejores ramas del árbol que representa el espacio de soluciones.
 - (b) ...un recorrido en profundidad del árbol que representa el espacio de soluciones.
 - (c) ...un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.
4. 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, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.
 - (b) Sí, tanto si se usan cotas optimistas para podar el árbol de búsqueda como si no.
 - (c) No, ya que en cualquier caso se deben explorar todas las soluciones factibles.

5. En los algoritmos de *ramificación y poda* ...

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

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

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

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

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

8. 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.

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

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

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

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

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

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

Di cuál de las siguientes afirmaciones es *cierta*:

- (a) Si $\mathbf{g}(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal en el caso peor del algoritmo de ordenación *quicksort*.
- (b) Si $\mathbf{g}(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación *quicksort*.
- (c) Si $\mathbf{g}(n) \in \Theta(1)$ la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación *mergesort*.

12. Sea la siguiente relación de recurrencia

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

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

- (a) $\mathbf{g}(n) = \log n$
- (b) $\mathbf{g}(n) = 1$
- (c) $\mathbf{g}(n) = n$

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

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

14. 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.

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

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

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

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

- (a) $\Omega(1)$
- (b) $\Omega(n)$
- (c) Esta función no tiene caso mejor.

17. Dada la relación de recurrencia:

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

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

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

18. 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 convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.
- (b) Se puede mejorar la eficiencia del algoritmo guardando en una tabla el valor devuelto para cada conjunto de parámetros de cada llamada cuando ésta se produce por primera vez.
- (c) Se puede mejorar la eficiencia del algoritmo definiendo de antemano el orden en el que se deben calcular las soluciones a los subproblemas y llenando una tabla en ese orden.

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

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

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

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

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

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

- (a) No.
- (b) Sí, en cualquier caso.
- (c) Sí, pero siempre que la complejidad temporal conjunta de las operaciones de descomposición del problema y la combinación de las soluciones sea polinómica.

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

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

24. 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) $O(f) = \Omega(f) \cap \Theta(f)$
- (c) $\Omega(f) = \Theta(f) \cap O(f)$

25. En el esquema de *vuelta atrás*, los mecanismos de poda basados en la mejor solución hasta el momento...

- (a) ... pueden eliminar soluciones parciales que son factibles.
- (b) ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- (c) Las otras dos opciones son ambas son verdaderas.

26. ¿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) $\bullet(\log n)$
- (b) $\bullet(\log^2 n)$
- (c) $\bullet(n)$

27. 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) $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$
 - (b) $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$
 - (c) $(4^{\log_2(n)}) \subset O(n) \subset O(2^n)$
28. ¿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.
29. ¿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 monedas).
30. En los algoritmos de *backtracking*, ¿Puede el valor de una cota pesimista ser mayor que el valor de una cota optimista? (se entiende que ambas cotas se aplican sobre el mismo nodo)
- (a) No, el valor de la cota pesimista de un nodo nunca puede ser superior al de la cota optimista de ese mismo nodo.
 - (b) En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
 - (c) En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
31. Se quiere ordenar d números distintos comprendidos entre 1 y n . Para ello se usa un array de n booleanos que se inicializan primero a *false*. A continuación se recorren los d números cambiando los valores del elemento del vector de booleanos correspondiente a su número a *true*. Por último se recorre el vector de booleanos escribiendo los índices de los elementos del vector de booleanos que son *true*. ¿Es este algoritmo más rápido (asintóticamente) que el *mergesort*?
- (a) Sólo si $d \log d > k n$ (donde k es una constante que depende de la implementación)
 - (b) Sí, ya que el *mergesort* es $O(n \log n)$ y este es $O(n)$
 - (c) No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.

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

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

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

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

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

34. Supongamos el algoritmo de ordenación *Mergesort* modificado de manera que, en lugar de dividir el vector en dos partes, se divide en tres. Posteriormente se combinan las soluciones parciales. ¿Cuál sería la complejidad temporal asintótica de la combinación de las soluciones parciales?

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

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

- (a) El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
- (b) El problema de la mochila discreta.
- (c) El problema de la asignación de coste mínimo de n tareas a n trabajadores cuando el coste de asignar la tarea i al trabajador j , c_{ij} está tabulado en una matriz.

Respostes per a la modalitat 1

1. Siga la relació de recurrència següent:

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

Si $T(n) \in O(n)$, en quin d'aquests tres casos ens podem trobar?

- (a) $g(n) = \log n$
- (b) $g(n) = 1$
- (c) $g(n) = n$

2. Quin d'aquests problemes té una solució eficient utilitzant *programació dinàmica*?

- (a) El problema de l'assignació de tasques.
 - (b) La motilla discreta sense restriccions addicionals.
 - (c) El problema del canvi (retornar una quantitat de diners amb el mínim nombre de monedes).
3. En els algorismes de *backtracking*, pot el valor d'una fita pessimista ser més gran que el valor d'una fita optimista? (s'entén que ambdues fites s'apliquen sobre el mateix node)

- (a) No, el valor de la fita pessimista d'un node mai pot ser superior al de la fita optimista d'aquest mateix node.
- (b) En general sí, si es tracta d'un problema de maximització, encara que en ocasions els dos valors poden coincidir.
- (c) En general sí, si es tracta d'un problema de minimització, encara que en ocasions els dos valors poden coincidir.

4. Per a què pot servir la fita pessimista d'un node de *ramificació i poda*?

- (a) Per actualitzar el valor de la millor solució fins al moment.
- (b) Per descartar el node si no és prometedor.
- (c) Per obtenir una fita optimista més precisa.

5. De les expressions següents, o bé dues són certes i una falsa, o bé dues són falses i una certa. Marca la que (en aquest sentit) és diferent a les altres dues.

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

6. Donada la relació de recurrència:

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

(on p i a són enters majors que 1 i $g(n) = n^k$), què ha d'ocórrer perquè es complisca $T(n) \in \Theta(n^k)$?

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

7. La relació de recurrència següent expressa la complexitat d'un algorisme recursiu, on $g(n)$ és una funció polinòmica:

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

Digues quina de les següents afirmacions és certa:

- (a) Si $g(n) \in \Theta(n)$ la relació de recurrència representa la complexitat temporal en el cas pitllor de l'algorisme de ordenació *quicksort*.
- (b) Si $g(n) \in \Theta(n)$ la relació de recurrència representa la complexitat temporal en el cas millor de l'algorisme de ordenació *quicksort*.
- (c) Si $g(n) \in \Theta(1)$ la relació de recurrència representa la complexitat temporal en el cas millor de l'algorisme d'ordenació *mergesort*.

8. Quina és la complexitat temporal de la següent funció?

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

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

9. Quina és la complexitat temporal en el millor dels casos de la següent funció?

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

- (a) $\Omega(1)$
- (b) $\Omega(n)$
- (c) Aquesta funció no té cas millor.

10. Davant un problema d'optimització resolt mitjançant *backtracking*, pot ocórrer que l'ús de les fites pessimistes i optimistes siga inútil, o fins i tot perjudicial?

- (a) Sí, ja que és possible que a pesar d'utilitzar aquestes fites no es descarte cap node.
- (b) Segons el tipus de fita; les pessimistes pot ser que no descarten cap node però l'ús de fites optimistes garanteix la reducció l'espai de recerca.
- (c) No, les fites tant optimistes com a pessimistes garanteixen la reducció de l'espai de solucions i per tant l'eficiència de l'algorisme.

11. Indiqueu quina d'aquestes afirmacions és *certa*.

- (a) La memoïtzació evita que un algorisme recursiu ingenu resolga repetidament el mateix problema.
- (b) L'avantatge de la solució de programació dinàmica iterativa al problema de la motxilla discreta és que mai es realitzen càlculs innecessaris.
- (c) Els algorismes iteratius de programació dinàmica utilitzen memoïtzació per evitar resoldre de nou els mateixos subproblemes que es tornen a presentar.

12. Per a quin d'aquests problemes d'optimització es coneix almenys una solució voraç?
- (a) L'arbre de recobriment mínim per a un graf no dirigit amb pesos.
- (b) El problema de la motxilla discreta.
- (c) El problema de l'assignació de cost mínim de n tasques a n treballadors quan el cost d'assignar la tasca i al treballador j , c_{ij} està tabulat en una matriu.
13. Garanteix l'ús d'una estratègia "divideix i venceràs" l'existència d'una solució de complexitat temporal polinòmica a qualsevol problema?
- (a) No
- (b) Sí, en qualsevol cas.
- (c) Sí, però sempre que la complexitat temporal conjunta de les operacions de descomposició del problema i la combinació de les solucions siga polinòmica.
14. La complexitat temporal de la solució via tornada arrere al problema de la motxilla discreta sense fraccionament és...
- (a) ... exponencial en el cas pitjor.
- (b) ... quadràtica en el cas pitjor.
- (c) ... exponencial en qualsevol cas.
15. Quin dels criteris següents proporcionaria una fita optimista per al problema de trobar el camí mes curt entre dues ciutats (se suposa que el graf és connex)?.
- (a) Calcular la distància recorreguda movent-se a l'atzar pel graf fins a arribar (per atzar) a la ciutat de destinació.
- (b) Calcular la distància geomètrica (en línia recta) entre les ciutats d'origen i de destinació.
- (c) Utilitzar la solució (subóptima) que s'obté quan es resol el problema mitjançant un algorisme voraç.
16. Es vol reduir la complexitat temporal de la següent funció fent ús de programació dinàmica. Quin seria la complexitat temporal resultant?

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

- (a) Es pot reduir fins a lineal.
- (b) Cuadràtica
- (c) La funció no compleix amb els requisits necessaris per a poder aplicar programació dinàmica.

17. Si $\lim_{n \rightarrow \infty} (f(n)/n^2) = k$, i $k \neq 0$, quina d'aquestes tres afirmacions és certa?
- (a) $f(n) \in \Theta(n^2)$
 - (b) $f(n) \in \Omega(n^3)$
 - (c) $f(n) \in \Theta(n^3)$
18. Siga A una matriu quadrada $n \times n$. Es tracta de buscar una permutació de les columnes tal que la suma dels elements de la diagonal de la matriu resultant siga mínima. Indiqueu quina de les següents afirmacions és certa.
- (a) La complexitat temporal de la millor solució possible al problema és $O(n \log n)$.
 - (b) La complexitat temporal de la millor solució possible al problema està en $\Omega(n^n)$.
 - (c) Si es construeix una solució al problema basada en l'esquema de ramificació i poda, una bona elecció de fites optimistes i pessimistes podria evitar l'exploració de totes les permutacions possibles.
19. De les expressions següents, o bé dues són certes i una falsa, o bé dues són falses i una certa. Marca la que (en aquest sentit) és diferent a les altres dues.
- (a) $\Theta(f) = O(f) \cap \Omega(f)$
 - (b) $O(f) = \Omega(f) \cap \Theta(f)$
 - (c) $\Omega(f) = \Theta(f) \cap O(f)$
20. De les expressions següents, o bé dues són certes i una falsa, o bé dues són falses i una certa. Marca la que (en aquest sentit) és diferent a les altres dues.
- (a) $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$
 - (b) $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$
 - (c) $(4^{\log_2(n)}) \subset O(n) \subset O(2^n)$
21. Quan es resol el problema de la motxilla discreta usant l'estrategia de tornada arrere, pot ocórrer que es tarde menys a trobar la solució óptima si es prova primer a ficar cada objecte abans de no ficar-ho?
- (a) Sí, però només si s'usen fites optimistes per podar l'arbre de recerca.
 - (b) Sí, tant si s'usen fites optimistes per podar l'arbre de recerca com si no.
 - (c) No, ja que en qualsevol cas s'han d'explorar totes les solucions factibles.

22. En els algorismes de *ramificació i poda* ...

- (a) Una fita optimista és necessàriament un valor insuperable; si no fóra així es podria podar el node que condueix a la solució òptima.
- (b) Una fita optimista és necessàriament un valor assolible; si no és així no està garantit que es trobe la solució òptima.
- (c) Una fita pessimista és el valor que com a màxim aconsegueix qualsevol node factible que no és l'òptim.

23. En un algorisme de *ramificació i poda*, l'ordre escollit per prioritzar els nodes en la llista de nodes vius ...

- (a) ... mai afecta al temps necessari per trobar la solució òptima.
- (b) ... determina la complexitat temporal en el pitjor dels casos de l'algorisme.
- (c) ... pot influir en el nombre de nodes que es descarten sense arribar a expandir-los.

24. Els algorismes de *tornada arrere* que fan ús de fites optimistes generen les solucions possibles al problema mitjançant ...

- (a) ... un recorregut guiat per les que poden ser les millors branques de l'arbre que representa l'espai de solucions.
- (b) ... un recorregut en profunditat de l'arbre que representa l'espai de solucions.
- (c) ... un recorregut guiat per una cua de prioritat d'on s'extrauen primer els nodes que representen els subarbres més prometedors de l'espai de solucions.

25. Què s'entén per *grandària del problema*?

- (a) La quantitat d'espai en memòria que es necessita per codificar una instància d'aquest problema.
- (b) El valor màxim que pot prendre una instància qualsevol d'aquest problema.
- (c) El nombre de paràmetres que componen el problema.

26. L'algorisme d'ordenació *Quicksort* divideix el problema en dos subproblems. Quina és la complexitat temporal asimptòtica de realitzar aquesta divisió?

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

27. Es vol ordenar d números diferents compresos entre 1 i n . Per a això s'usa un array de n booleans que s'inicialitzen primer a *false*. A continuació es recorren els d números canviant els valors de l'element del vector de booleans corresponent al seu números a *true*. Finalment es recorre el vector de booleans escrivint els índexs dels elements del vector de booleans que són *true*. És aquest algorisme més ràpid (asimptòticament) que el *mergesort*?
- (a) Només si $d \log d > kn$ (on k és una constant que depén de la implementació)
 - (b) Sí, ja que el *mergesort* és $O(n \log n)$ i aquest és $O(n)$
 - (c) No, ja que aquest algorisme ha de recórrer diverses vegades el vector de booleans.
28. En el problema de l'acolorit de grafs (mínim nombre de colors necessaris per a acolorir tots els vèrtexs d'un graf de manera que no queden dos adjacents amb el mateix color) resolt mitjançant *ramificació i poda*, de quina manera s'hauria d'ordenar la llista de nodes vius per a obtenir una solució acceptable?
- (a) Per cota optimista: explorant primer els nodes amb menor cota optimista.
 - (b) Per cota pessimista: explorant primer els nodes amb menor cota optimista.
 - (c) les altres dues opcions poden ser ambdues correctes.
29. Quina és la definició correcta de $\Omega(g)$?
- (a) $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ | \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \geq cg(n)\}$
 - (b) $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ | \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$
 - (c) $\Omega(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ | \forall c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$
30. La solució recursiva ingènua (però correcta) a un problema d'optimització crida més d'una vegada a la funció amb els mateixos paràmetres. Una de les tres afirmacions següents és falsa.
- (a) Es pot millorar l'eficiència de l'algorisme convertint l'algorisme recursiu directament en iteratiu sense canviar el seu funcionament bàsic.
 - (b) Es pot millorar l'eficiència de l'algorisme guardant en una taula el valor retornat per a cada conjunt de paràmetres de cada crida quan aquesta es produeix per primera vegada.
 - (c) Es pot millorar l'eficiència de l'algorisme definint per endavant l'ordre en el qual s'han de calcular les solucions als subproblemes i omplint una taula en aquest ordre.

31. En l'esquema de *tornada enrere*, els mecanismes de poda basats en la millor solució fins al moment...
- (a) ... poden eliminar solucions parcials que són factibles.
 - (b) ... garanteixen que no s'explorarà mai tot l'espai de solucions possibles.
 - (c) Les altres dues opcions són ambdues certes.
32. En el problema de l'acolorit de grafs (mínim nombre de colors necessaris per a acolorir tots els vèrtexs d'un graf de manera que no queden dos adjacents amb el mateix color) resolt mitjançant *ramificació i poda*, una cota optimista és el resultat de l'assumir que ...
- (a) ... no es van a utilitzar colors diferents als ja utilitzats.
 - (b) ... es van a utilitzar tants colors diferents als ja utilitzats com a vèrtexs queden per acolorir.
 - (c) ... només vaa ser necessari un color més.
33. Considerem l'algorisme d'ordenació *Mergesort* modificat de manera que, en comptes de dividir el vector en dues parts, es divideix en tres. Posteriorment es combinen les solucions parcials. Quina seria la complexitat temporal asimptòtica de la combinació de les solucions parcials?
- (a) Cap de les altres dues opcions és certa.
 - (b) $\Theta(n)$
 - (c) $\Theta(\log_3 n)$
34. Es vol reduir la complexitat temporal de la funció g usant programació dinàmica iterativa. Quina seria la complexitat espacial de l'algorisme resultant?
- ```
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) Quadràtica
  - (b) Lineal
  - (c) Cúbica
35. Si  $f \in \Theta(g_1)$  y  $f \in \Theta(g_2)$  aleshores
- (a)  $f \notin \Theta(\max(g_1, g_2))$
  - (b)  $f \in \Theta(g_1 \cdot g_2)$
  - (c)  $f \in \Theta(g_1 + g_2)$

## Respuestas para la modalidad 3

1. Sea la siguiente relación de recurrencia

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

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

- (a)  $g(n) = \log n$
- (b)  $g(n) = 1$
- (c)  $g(n) = n$

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

- (a) El problema de la asignación de tareas.
  - (b) La mochila discreta sin restricciones adicionales.
  - (c) El problema del cambio (devolver una cantidad de dinero con el mínimo número de monedas).
3. En los algoritmos de *backtracking*, ¿Puede el valor de una cota pesimista ser mayor que el valor de una cota optimista? (se entiende que ambas cotas se aplican sobre el mismo nodo)
- (a) No, el valor de la cota pesimista de un nodo nunca puede ser superior al de la cota optimista de ese mismo nodo.
  - (b) En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
  - (c) En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.

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

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

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

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

6. Dada la relación de recurrencia:

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

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

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

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

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

Di cuál de las siguientes afirmaciones es *cierta*:

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

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

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

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

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

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

- (a)  $\Omega(1)$
- (b)  $\Omega(n)$
- (c) Esta función no tiene caso mejor.

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

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

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

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

12. ¿Para cuál de estos problemas de optimización se conoce al menos una solución voraz óptima?
- (a) El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
  - (b) El problema de la mochila discreta.
  - (c) El problema de la asignación de coste mínimo de  $n$  tareas a  $n$  trabajadores cuando el coste de asignar la tarea  $i$  al trabajador  $j$ ,  $c_{ij}$  está tabulado en una matriz.
13. Garantiza el uso de una estrategia “divide y vencerás” la existencia de una solución de complejidad temporal polinómica a cualquier problema?
- (a) No.
  - (b) Sí, en cualquier caso.
  - (c) Sí, pero siempre que la complejidad temporal conjunta de las operaciones de descomposición del problema y la combinación de las soluciones sea polinómica.
14. La complejidad temporal de la solución de *vuelta atrás* al problema de la mochila discreta es ...
- (a) ...exponencial en el caso peor..
  - (b) ...cuadrática en el caso peor.
  - (c) ...exponencial en cualquier caso.
15. Cuál de los siguientes criterios proporcionaría una cota optimista para el problema de encontrar el camino mas corto entre dos ciudades (se supone que el grafo es conexo).
- (a) Calcular la distancia recorrida moviéndose al azar por el grafo hasta llegar (por azar) a la ciudad destino.
  - (b) Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.
  - (c) Utilizar la solución (subóptima) que se obtiene al resolver el problema mediante un algoritmo voraz.
16. Se quiere reducir la complejidad temporal de la siguiente función haciendo uso de programación dinámica. ¿Cuál sería la complejidad temporal resultante?
- ```
unsigned g( unsigned n, unsigned r){
    if (r==0 || r==n)
        return 1;
    return g(n-1, r-1) + g(n-1, r);
}
```
- (a) Se puede reducir hasta lineal.
 - (b) Cuadrática
 - (c) La función no cumple con los requisitos necesarios para poder aplicar programación dinámica.

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

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

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

- (a) La complejidad temporal de la mejor solución posible al problema es $O(n \log n)$.
- (b) La complejidad temporal de la mejor solución posible al problema está en $\Omega(n^n)$.
- (c) Si se construye una solución al problema basada en el esquema de ramificación y poda, una buena elección de cotas optimistas y pesimistas podría evitar la exploración de todas las permutaciones posibles.

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

- (a) $\Theta(f) = O(f) \cap \Omega(f)$
- (b) $O(f) = \Omega(f) \cap \Theta(f)$
- (c) $\Omega(f) = \Theta(f) \cap O(f)$

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

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

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

- (a) Sí, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.
- (b) Sí, tanto si se usan cotas optimistas para podar el árbol de búsqueda como si no.
- (c) No, ya que en cualquier caso se deben explorar todas las soluciones factibles.

22. En los algoritmos de *ramificación y poda* ...

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

23. En un algoritmo de *ramificación y poda*, el orden escogido para priorizar los nodos en la lista de nodos vivos ...

- (a) ... nunca afecta al tiempo necesario para encontrar la solución óptima.
- (b) ... determina la complejidad temporal en el peor de los casos del algoritmo.
- (c) ... puede influir en el número de nodos que se descartan sin llegar a expandirlos.

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

- (a) ... un recorrido guiado por estimaciones de las que pueden ser las mejores ramas del árbol que representa el espacio de soluciones.
- (b) ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- (c) ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.

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

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

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

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

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

- (a) Sólo si $d \log d > k n$ (donde k es una constante que depende de la implementación)
- (b) Sí, ya que el *mergesort* es $O(n \log n)$ y este es $O(n)$
- (c) No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.

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

- (a) Por cota optimista: explorando primero los nodos con menor cota optimista.
- (b) Por cota pesimista: explorando primero los nodos con menor cota pesimista.
- (c) las otras dos opciones pueden ser ambas correctas.

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

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

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

- (a) Se puede mejorar la eficiencia del algoritmo convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.
- (b) Se puede mejorar la eficiencia del algoritmo guardando en una tabla el valor devuelto para cada conjunto de parámetros de cada llamada cuando ésta se produce por primera vez.
- (c) Se puede mejorar la eficiencia del algoritmo definiendo de antemano el orden en el que se deben calcular las soluciones a los subproblemas y llenando una tabla en ese orden.

31. En el esquema de *vuelta atrás*, los mecanismos de poda basados en la mejor solución hasta el momento...

- (a) ... pueden eliminar soluciones parciales que son factibles.
- (b) ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- (c) Las otras dos opciones son ambas son verdaderas.

32. En el problema del coloreado de grafos (mínimo número de colores necesarios para colorear todos los vértices de un grafo de manera que no queden dos adyacentes con el mismo color) resuelto mediante *ramificación y poda*, una cota optimista es el resultado de asumir que ...

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

33. Supongamos el algoritmo de ordenación *Mergesort* modificado de manera que, en lugar de dividir el vector en dos partes, se divide en tres. Posteriormente se combinan las soluciones parciales. ¿Cuál sería la complejidad temporal asintótica de la combinación de las soluciones parciales?

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

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

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

- (a) Cuadrática.
- (b) Lineal.
- (c) Cúbica.

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

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

Análisis y Diseño de Algoritmos
5 de julio de 2012
Examen final (duración: 120 min.)

Instrucciones:

- No podéis consultar ningún material ni hablar con nadie.
- Poned vuestros datos en la hoja de respuestas de lectura óptica que se os entregará.
- **No olvidéis indicar la modalidad del examen en la hoja de respuestas.**
- Elegid en cada pregunta la opción que creáis correcta, y marcadla cómo se indica en la hoja de respuestas.
- La nota del examen se calcula así:

$$\text{nota} = 10 \times (\text{aciertos} - \frac{1}{2}\text{errores})/\text{preguntas}$$

- Las respuestas en blanco ni restan ni suman puntos.
- Hay que entregar tanto las hojas de preguntas como la de respuestas (aunque las hojas de preguntas las podéis marcar).
- Tenéis 120 min. para hacer el examen.
- Si tenéis alguna duda, levantad la mano y un profesor irá a atenderos.

Modalidad 2

Preguntas:

1. Los algoritmos de ordenación *Quicksort* y *Mergesort* tienen en común...
 - (a) ... que ordenan el vector sin usar espacio adicional.
 - (b) ... que se ejecutan en tiempo $O(n)$.
 - (c) ... que aplican la estrategia de *divide y vencerás*.
2. ¿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 coste asintótico en el caso peor.
 - (b) El hecho que la solución de *ramificación y poda* puede empezar con una solución subóptima voraz y la de *vuelta atrás* no.
 - (c) El orden de exploración de las soluciones.

3. Tenemos un conjunto de n enteros positivos y queremos encontrar el subconjunto de tamaño m de suma mínima.
- (a) Lo más adecuado sería usar una técnica de ramificación y poda, aunque en el peor caso el coste temporal asintótico (o complejidad temporal) sería exponencial.
 - (b) Para encontrar la solución habría que probar con todas las combinaciones posibles de m enteros, con lo que la técnica de ramificación y poda no aporta nada con respecto a vuelta atrás.
 - (c) Una técnica voraz daría una solución óptima.
4. Di cuál de estos resultados de coste temporal asintótico es falsa:
- (a) La ordenación de un vector usando el algoritmo *Quicksort* requiere en el peor caso un tiempo en $\Omega(n^2)$.
 - (b) La ordenación de un vector usando el algoritmo *Mergesort* requiere en el peor caso un tiempo en $\Omega(n^2)$.
 - (c) La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en $O(\log n)$.
5. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
- (a) 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.
 - (b) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
 - (c) Lo más importante es conseguir una cota pesimista muy adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.
6. La complejidad temporal (o coste temporal asintótico) en el mejor de los casos...
- (a) ... es una función de la talla, o tamaño del problema, que tiene que estar definida para todos los posibles valores de ésta.
 - (b) ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.
 - (c) Las dos anteriores son verdaderas.

7. Tenemos n substancias diferentes en polvo y queremos generar todas las distintas formas de mezclarlas de forma que el peso total no supere un gramo. Como la balanza que tenemos solo tiene una precisión de 0.1 gramos, no se considerarán pesos que no sean múltiplos de esta cantidad. Queremos hacer un programa que genere todas las combinaciones posibles.
- (a) 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.
8. Un algoritmo recursivo basado en el esquema *divide y vencerás*...
- (~~a~~) ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a .
 - (b) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.
 - (c) Las dos anteriores son verdaderas.
9. Dado un problema de optimización, ¿cuándo se puede aplicar el método de vuelta atrás?
- (~~a~~) Es condición necesaria (aunque no suficiente) que el dominio de las decisiones sea discreto o discretizable.
 - (b) Es condición necesaria y suficiente que el dominio de las decisiones sea discreto o discretizable.
 - (c) No sólo es condición necesaria que el dominio de las decisiones sea discreto o discretizable; además, debe cumplirse que se puedan emplear mecanismos de poda basados en la mejor solución hasta el momento.
10. ¿Cuál de estas tres expresiones es cierta?
- (~~a~~) $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$
 - (~~b~~) $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
 - (c) $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$
11. Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2) + n$; $f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:
- (a) $f(n) \in \Theta(n^2)$
 - (~~b~~) $f(n) \in \Theta(n \log n)$
 - (c) $f(n) \in \Theta(n)$
12. Indicad cuál de estas tres expresiones es falsa:
- (a) $\Theta(n/2) = \Theta(n)$
 - (b) $\Theta(n) \subseteq O(n)$
 - (~~c~~) $\Theta(n) \subseteq \Theta(n^2)$

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

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

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

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

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

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

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

16. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

- (a) ... pueden eliminar vectores que representan posibles soluciones factibles.
(b) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.
(c) Las dos anteriores son verdaderas.

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

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

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

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

19. Las relaciones de recurrencia...

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

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

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

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

- (a) $O(n \log(n))$
 - (b) $O(n^2)$
 - (c) $O(2^n)$
21. La versión de *Quicksort* que utiliza como pivote el elemento del vector que ocupa la posición central...

- (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.

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

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

- (a) $O(1)$
 - (b) $O(\log(n))$
 - (c) $O(n)$
23. 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 mínimas monedas.
- (c) El problema del viajante de comercio

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

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

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

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

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

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

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

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

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

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

29. ¿Cuál de estas tres expresiones es falsa?

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

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

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

31. El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continue ordenado es...

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

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

33. El coste temporal asintótico del programa

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

y el del programa

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

son...

- (a) ... el del primero, menor que el del segundo.
- (b) ... el del segundo, menor que el del primero.
- ~~(c)~~ ... iguales.

34. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?

- (a) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
- (b) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
- ~~(c)~~ Vuelta atrás, es el esquema más eficiente para este problema.

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

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

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

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

37. 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)** Sí.
- (b) No, ya que el índice del almacén sería un número real y no entero.
- (c) No, ya que no podríamos almacenar los resultados intermedios en el almacén.

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

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

39. Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?

- (a) un vector de booleanos.
- (b) un par de enteros que indiquen los cortes realizados y el valor acumulado.
- (c)** una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.

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

Análisis y Diseño de Algoritmos
5 de julio de 2012
Examen final (duración: 120 min.)

Instrucciones:

- No podéis consultar ningún material ni hablar con nadie.
- Poned vuestros datos en la hoja de respuestas de lectura óptica que se os entregará.
- **No olvidéis indicar la modalidad del examen en la hoja de respuestas.**
- Elegid en cada pregunta la opción que creáis correcta, y marcadla cómo se indica en la hoja de respuestas.
- La nota del examen se calcula así:

$$\text{nota} = 10 \times (\text{aciertos} - \frac{1}{2}\text{errores})/\text{preguntas}$$

- Las respuestas en blanco ni restan ni suman puntos.
- Hay que entregar tanto las hojas de preguntas como la de respuestas (aunque las hojas de preguntas las podéis marcar).
- Tenéis 120 min. para hacer el examen.
- Si tenéis alguna duda, levantad la mano y un profesor irá a atenderos.

Modalidad 2

Preguntas:

1. Los algoritmos de ordenación *Quicksort* y *Mergesort* tienen en común...
 - (a) ... que ordenan el vector sin usar espacio adicional.
 - (b) ... que se ejecutan en tiempo $O(n)$.
 - (c) ... que aplican la estrategia de *divide y vencerás*.
2. ¿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 coste asintótico en el caso peor.
 - (b) El hecho que la solución de *ramificación y poda* puede empezar con una solución subóptima voraz y la de *vuelta atrás* no.
 - (c) El orden de exploración de las soluciones.

3. Tenemos un conjunto de n enteros positivos y queremos encontrar el subconjunto de tamaño m de suma mínima.
 - (a) Lo más adecuado sería usar una técnica de ramificación y poda, aunque en el peor caso el coste temporal asintótico (o complejidad temporal) sería exponencial.
 - (b) Para encontrar la solución habría que probar con todas las combinaciones posibles de m enteros, con lo que la técnica de ramificación y poda no aporta nada con respecto a vuelta atrás.
 - (c) Una técnica voraz daría una solución óptima.
4. Di cuál de estos resultados de coste temporal asintótico es falsa:
 - (a) La ordenación de un vector usando el algoritmo *Quicksort* requiere en el peor caso un tiempo en $\Omega(n^2)$.
 - (b) La ordenación de un vector usando el algoritmo *Mergesort* requiere en el peor caso un tiempo en $\Omega(n^2)$.
 - (c) La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en $O(\log n)$.
5. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
 - (a) 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.
 - (b) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
 - (c) Lo más importante es conseguir una cota pesimista muy adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.
6. La complejidad temporal (o coste temporal asintótico) en el mejor de los casos...
 - (a) ... es una función de la talla, o tamaño del problema, que tiene que estar definida para todos los posibles valores de ésta.
 - (b) ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.
 - (c) Las dos anteriores son verdaderas.

7. Tenemos n substancias diferentes en polvo y queremos generar todas las distintas formas de mezclarlas de forma que el peso total no supere un gramo. Como la balanza que tenemos solo tiene una precisión de 0.1 gramos, no se considerarán pesos que no sean múltiplos de esta cantidad. Queremos hacer un programa que genere todas las combinaciones posibles.
- No hay ningún problema en usar una técnica de vuelta atrás.
 - No se puede usar vuelta atrás porque las decisiones no son valores discretos.
 - No se puede usar vuelta atrás porque el número de combinaciones es infinito.
8. Un algoritmo recursivo basado en el esquema *divide y vencerás*...
- ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a .
 - ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.
 - Las dos anteriores son verdaderas.
9. Dado un problema de optimización, ¿cuándo se puede aplicar el método de vuelta atrás?
- Es condición necesaria (aunque no suficiente) que el dominio de las decisiones sea discreto o discretizable.
 - Es condición necesaria y suficiente que el dominio de las decisiones sea discreto o discretizable.
 - No sólo es condición necesaria que el dominio de las decisiones sea discreto o discretizable; además, debe cumplirse que se puedan emplear mecanismos de poda basados en la mejor solución hasta el momento.
10. ¿Cuál de estas tres expresiones es cierta?
- $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$
 - $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
 - $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$
11. Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2) + n$; $f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:
- $f(n) \in \Theta(n^2)$
 - $f(n) \in \Theta(n \log n)$
 - $f(n) \in \Theta(n)$
12. Indicad cuál de estas tres expresiones es falsa:
- $\Theta(n/2) = \Theta(n)$
 - $\Theta(n) \subseteq O(n)$
 - $\Theta(n) \subseteq \Theta(n^2)$

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

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

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

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

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

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

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

16. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

- (a) ... pueden eliminar vectores que representan posibles soluciones factibles.
- (b) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.
- (c) Las dos anteriores son verdaderas.

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

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

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

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

19. Las relaciones de recurrencia...

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

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

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

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

- (a) $O(n \log(n))$
 - (b) $O(n^2)$
 - (c) $O(2^n)$
21. La versión de *Quicksort* que utiliza como pivote el elemento del vector que ocupa la posición central...

- (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.

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

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

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

23. 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 mínimas monedas.
- (c) El problema del viajante de comercio

24. ¿Qué algoritmo es asintóticamente más rápido, el *quicksort* o el *mergesort*?
- (a) como su nombre indica, el *quicksort*.
 - (b) son los dos igual de rápidos, ya que el coste temporal asintótico de ambos es $O(n \log(n))$.
 - (c) el *mergesort* es siempre más rápido o igual (salvo una constante) que el *quicksort*.
25. El coste temporal del algoritmo de ordenación por inserción es...
- (a) ... $O(n^2)$.
 - (b) ... $O(n)$.
 - (c) ... $O(n \log n)$.
26. Los algoritmos de programación dinámica hacen uso...
- (a) ...de que la solución óptima se puede construir añadiendo el componente óptimo de los restantes, uno a uno.
 - (b) ...de que se puede ahorrar esfuerzo guardando los resultados de esfuerzos anteriores.
 - (c) ...de una estrategia trivial consistente en examinar todas las soluciones posibles.
27. ¿Cuál de estos tres problemas de optimización no tiene una solución voraz (*greedy*) que sea óptima?
- (a) El problema de la mochila continua o con fraccionamiento.
 - (b) El problema de la mochila discreta.
 - (c) El árbol de cobertura de coste mínimo de un grafo conexo.
28. El problema de la función compuesta mínima consiste en encontrar, a partir de un conjunto de funciones dadas, la secuencia mínima de composiciones de éstas que permita transformar un número n en otro m . Se quiere resolver mediante ramificación y poda. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?
- (a) Mediante un vector de booleanos.
 - (b) Mediante un vector de reales.
 - (c) Este problema no se puede resolver usando ramificación y poda si no se fija una cota superior al número total de aplicaciones de funciones.
29. ¿Cuál de estas tres expresiones es falsa?
- (a) $2n^2 + 3n + 1 \in O(n^3)$
 - (b) $n + n \log(n) \in \Omega(n)$
 - (c) $n + n \log(n) \in \Theta(n)$

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

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

31. El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continue ordenado es...

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

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

33. El coste temporal asintótico del programa

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

y el del programa

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

son...

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

34. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?

- (a) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
- (b) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
- (c) Vuelta atrás, es el esquema más eficiente para este problema.

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

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

36. La solución recursiva ingenua a un determinado problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en...
- (a) ... un algoritmo del estilo de *divide y vencerás*.
 - (b) ... un algoritmo de programación dinámica.
 - (c) ... un algoritmo voraz.
37. 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) Sí.
  - (b) No, ya que el índice del almacén sería un número real y no entero.
  - (c) No, ya que no podríamos almacenar los resultados intermedios en el almacén.
38. Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:
- (a) El valor de la mochila continua correspondiente .
  - (b) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
  - (c) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
39. Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?
- (a) un vector de booleanos.
  - (b) un par de enteros que indiquen los cortes realizados y el valor acumulado.
  - (c) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.

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

Algoritmia - Grado en Ingeniería Robótica  
Simulacro de examen final  
Convocatoria C2.

**Instrucciones:**

- No se puede consultar ningún material.
- El examen consta de 40 preguntas de tres opciones. Todas puntúan igual.
- Solo una opción se considera correcta. Marca la que consideres más apropiada.
- Cada pregunta mal contestada resta media pregunta bien contestada.
- Las preguntas sin contestar ni restan ni suman puntos.
- Dispones de 60 min. para hacer el examen.

**Modalidad 1**

**Preguntas:**

1. Si el coste temporal de un algoritmo es  $T(n)$ , ¿cuál de las siguientes situaciones es imposible?
  - (a)  $T(n) \in O(n)$  y  $T(n) \in \Theta(n)$
  - (b)  $T(n) \in \Omega(n)$  y  $T(n) \in \Theta(n^2)$
  - (c)  $T(n) \in \Theta(n)$  y  $T(n) \in \Omega(n^2)$
2. ¿Cuál es el coste espacial asintótico del siguiente algoritmo?

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

  - (a)  $O(1)$
  - (b)  $O(\log(n))$
  - (c)  $O(n)$
3. ¿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) El coste temporal promedio.
  - (b) El coste temporal asintótico en el caso medio.
  - (c) Nada de interés.

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

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

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

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

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

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

6. La función  $\gamma$  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) Sí.
- (b) No, ya que el índice del almacén sería un número real y no entero.
- (c) No, ya que no podríamos almacenar los resultados intermedios en el almacén.

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

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

8. 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.
9. Tenemos un conjunto de  $n$  enteros positivos y queremos encontrar el subconjunto de tamaño  $m$  de suma mínima ( $m \leq n$ ).
- (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.
10. Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?
- (a) un vector de booleanos.
  - (b) un par de enteros que indiquen los cortes realizados y el valor acumulado.
  - (c) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.
11. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
- (a) 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.
  - (b) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
  - (c) Lo más importante es conseguir una cota pesimista muy adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.

12. El problema de la función compuesta mínima consiste en encontrar, a partir de un conjunto de funciones dadas, la secuencia mínima de composiciones de éstas que permita transformar un número  $n$  en otro  $m$ . Se quiere resolver mediante ramificación y poda. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?
- (a) Mediante un vector de booleanos.
  - (b) Mediante un vector de reales.
  - (c) Este problema no se puede resolver usando ramificación y poda si no se fija una cota superior al número total de aplicaciones de funciones.
13. ¿Cuál de estas tres expresiones es falsa?
- (a)  $2n^2 + 3n + 1 \in O(n^3)$
  - (b)  $n + n \log(n) \in \Omega(n)$
  - (c)  $n + n \log(n) \in \Theta(n)$
14. Indica cuál es el coste temporal asintótico (o complejidad temporal), en función de  $n$ , del programa siguiente:
- ```
s=0; for (i=0; i<n; i++) for (j=i; j<n; j++) s+=n*i+j;
```
- (a) Es $O(n^2)$ pero no $\Omega(n^2)$.
 - (b) Es $\Theta(n^2)$
 - (c) Es $\Theta(n)$
15. El coste temporal asintótico del programa
- ```
s=0; for (i=0; i<n; i++) for (j=i; j<n; j++) s+=i*j;
```
- y el del programa
- ```
s=0; for (i=0; i<n; i++) for (j=0; j<n; j++) s+=i*i*j;
```
- son...
- (a) ... el del primero, menor que el del segundo.
 - (b) ... el del segundo, menor que el del primero.
 - (c) ... iguales.
16. La solución recursiva ingenua a un determinado problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en...
- (a) ... un algoritmo del estilo de *divide y vencerás*.
 - (b) ... un algoritmo de programación dinámica.
 - (c) ... un algoritmo voraz.

17. ¿Cuál de estos tres problemas de optimización no tiene una solución voraz (*greedy*) que sea óptima?
- El problema de la mochila continua o con fraccionamiento.
 - El problema de la mochila discreta.
 - El árbol de cobertura de coste mínimo de un grafo conexo.
18. ¿Cuál es la diferencia principal entre una solución de *vuelta atrás* y una solución de *ramificación y poda* para el problema de la mochila?
- El coste asintótico en el caso peor.
 - El hecho que la solución de *ramificación y poda* puede empezar con una solución subóptima voraz y la de *vuelta atrás* no.
 - El orden de exploración de las soluciones.
19. Las relaciones de recurrencia...
- ... aparecen únicamente cuando la solución son del tipo *divide y vencerás*.
 - ... expresan recursivamente el coste temporal o espacial de un algoritmo.
 - ... sirven para reducir el coste temporal de una solución cuando este es prohibitivo.
20. Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n - 1) + 1$; $f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:
- $f(n) \in \Theta(n^2)$
 - $f(n) \in \Theta(2^n)$
 - $f(n) \in \Theta(n)$
21. Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2) + n$; $f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:
- $f(n) \in \Theta(n^2)$
 - $f(n) \in \Theta(n \log n)$
 - $f(n) \in \Theta(n)$
22. Indicad cuál de estas tres expresiones es falsa:
- $\Theta(n/2) = \Theta(n)$
 - $\Theta(n) \subseteq O(n)$
 - $\Theta(n) \subseteq \Theta(n^2)$
23. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...
- ... no se puede usar para resolver problemas de optimización.
 - ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.
 - ... debe recorrer siempre todo el árbol.

24. Los algoritmos de ordenación *Quicksort* y *Mergesort* tienen en común...
- ... que ordenan el vector sin usar espacio adicional.
 - ... que se ejecutan en tiempo $O(n)$.
 - ... que aplican la estrategia de *divide y vencerás*.
25. Di cuál de estos resultados de coste temporal asintótico es falsa:
- La ordenación de un vector usando el algoritmo *Quicksort* requiere en el peor caso un tiempo en $\Omega(n^2)$.
 - La ordenación de un vector usando el algoritmo *Mergesort* requiere en el peor caso un tiempo en $\Omega(n^2)$.
 - La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en $O(\log n)$.
26. El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continúe ordenado es...
- $O(n)$.
 - $O(\log n)$.
 - $\Omega(n^2)$.
27. El coste temporal del algoritmo de ordenación por selección es...
- $O(n^2)$.
 - $O(n)$.
 - $O(n \log n)$.
28. Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:
- El valor de la mochila continua correspondiente .
 - El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
 - El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
29. Los algoritmos de programación dinámica hacen uso...
- ...de que la solución óptima se puede construir añadiendo el componente óptimo de los restantes, uno a uno.
 - ...de que se puede ahorrar esfuerzo guardando los resultados de esfuerzos anteriores.
 - ...de una estrategia trivial consistente en examinar todas las soluciones posibles.
30. Un programa con dos bucles anidados uno dentro del otro, cada uno de los cuales hace aproximadamente n iteraciones, tarda un tiempo
- $O(n^2)$
 - $O(2^n)$
 - $O(n)$

31. 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 mínimas monedas.
 - (c) El problema del viajante de comercio
32. ¿Cuál de estas tres expresiones es cierta?
- (a) $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$
 - (b) $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
 - (c) $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$
33. La versión de *Quicksort* que utiliza como pivote el elemento del vector que ocupa la posición central...
- (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.
34. Un algoritmo recursivo basado en el esquema *divide y vencerás*...
- (a) ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a .
 - (b) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.
 - (c) Las dos anteriores son verdaderas.
35. La complejidad temporal (o coste temporal asintótico) en el mejor de los casos...
- (a) ... es una función de la talla, o tamaño del problema, que tiene que estar definida para todos los posibles valores de ésta.
 - (b) ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.
 - (c) Las dos anteriores son verdaderas.
36. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...
- (a) ... pueden eliminar vectores que representan posibles soluciones factibles.
 - (b) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.
 - (c) Las dos anteriores son verdaderas.

37. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?
- (a) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
 - (b) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
 - (c) Vuelta atrás, es el esquema más eficiente para este problema.
38. 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) Divide y vencerás.
 - (b) Programación dinámica.
 - (c) Ramificación y poda.
39. La eficiencia de los algoritmos voraces se basa en...
- (a) ... el hecho de que, con antelación, las posibles decisiones se ordenan de mejor a peor.
 - (b) ... el hecho de que las decisiones tomadas no se reconsideran.
 - (c) En el esquema voraz no se puede hablar de eficiencia puesto que a menudo no resuelve el problema.
40. 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.

Respuestas para la modalidad 1

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

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

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

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

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

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

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

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

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

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

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

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

6. La función γ de un número semientero positivo (un número es semiente-
ro 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) Sí.
 - (b) No, ya que el índice del almacén sería un número real y no entero.
 - (c) No, ya que no podríamos almacenar los resultados intermedios en el almacén.
7. Pertenece $3n^2 + 3$ a $O(n^3)$?
- (a) No.
 - (b) Sólo para $c = 1$ y $n_0 = 5$
 - (c) Sí.
8. 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.
9. Tenemos un conjunto de n enteros positivos y queremos encontrar el subconjunto de tamaño m de suma mínima ($m \leq n$).
- (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.

10. Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?
- (a) un vector de booleanos.
 - (b) un par de enteros que indiquen los cortes realizados y el valor acumulado.
 - (c) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.
11. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
- (a) 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.
 - (b) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
 - (c) Lo más importante es conseguir una cota pesimista muy adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.
12. El problema de la función compuesta mínima consiste en encontrar, a partir de un conjunto de funciones dadas, la secuencia mínima de composiciones de éstas que permita transformar un número n en otro m . Se quiere resolver mediante ramificación y poda. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?
- (a) Mediante un vector de booleanos.
 - (b) Mediante un vector de reales.
 - (c) Este problema no se puede resolver usando ramificación y poda si no se fija una cota superior al número total de aplicaciones de funciones.
13. ¿Cuál de estas tres expresiones es falsa?
- (a) $2n^2 + 3n + 1 \in O(n^3)$
 - (b) $n + n \log(n) \in \Omega(n)$
 - (c) $n + n \log(n) \in \Theta(n)$
14. Indica cuál es el coste temporal asintótico (o complejidad temporal), en función de n , del programa siguiente:
- ```
s=0; for(i=0; i<n; i++) for(j=i; j<n; j++) s+=n*i+j;
```
- (a) Es  $O(n^2)$  pero no  $\Omega(n^2)$ .
  - (b) Es  $\Theta(n^2)$
  - (c) Es  $\Theta(n)$

15. El coste temporal asintótico del programa

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

y el del programa

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

son...

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

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

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

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

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

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

19. Las relaciones de recurrencia...

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

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

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

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

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

22. Indicad cuál de estas tres expresiones es falsa:

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

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

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

24. 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*.

25. Di cuál de estos resultados de coste temporal asintótico es falsa:

- (a) La ordenación de un vector usando el algoritmo *Quicksort* requiere en el peor caso un tiempo en  $\Omega(n^2)$ .  
 (b) La ordenación de un vector usando el algoritmo *Mergesort* requiere en el peor caso un tiempo en  $\Omega(n^2)$ .  
(c) La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en  $O(\log n)$ .

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

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

27. El coste temporal del algoritmo de ordenación por selección es...
- (a) ... $O(n^2)$ .
  - (b) ... $O(n)$ .
  - (c) ... $O(n \log n)$ .
28. Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:
- (a) El valor de la mochila continua correspondiente .
  - (b) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
  - (c) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
29. Los algoritmos de programación dinámica hacen uso...
- (a) ...de que la solución óptima se puede construir añadiendo el componente óptimo de los restantes, uno a uno.
  - (b) ...de que se puede ahorrar esfuerzo guardando los resultados de esfuerzos anteriores.
  - (c) ...de una estrategia trivial consistente en examinar todas las soluciones posibles.
30. Un programa con dos bucles anidados uno dentro del otro, cada uno de los cuales hace aproximadamente  $n$  iteraciones, tarda un tiempo
- (a)  $O(n^2)$
  - (b)  $O(2^n)$
  - (c)  $O(n)$
31. 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 mínimas monedas.
  - (c) El problema del viajante de comercio
32. ¿Cuál de estas tres expresiones es cierta?
- (a)  $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$
  - (b)  $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
  - (c)  $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$
33. La versión de *Quicksort* que utiliza como pivote el elemento del vector que ocupa la posición central...
- (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.

34. Un algoritmo recursivo basado en el esquema *divide y vencerás*...
- (a) ... alcanza su máxima eficiencia cuando el problema de tamaño  $n$  se divide en  $a$  problemas de tamaño  $n/a$ .  
(b) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.  
(c) Las dos anteriores son verdaderas.
35. La complejidad temporal (o coste temporal asintótico) en el mejor de los casos...
- (a) ... es una función de la talla, o tamaño del problema, que tiene que estar definida para todos los posibles valores de ésta.  
(b) ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.  
(c) Las dos anteriores son verdaderas.
36. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...
- (a) ... pueden eliminar vectores que representan posibles soluciones factibles.  
(b) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.  
(c) Las dos anteriores son verdaderas.
37. Se desea obtener todas las permutaciones de una lista compuesta por  $n$  elementos. ¿Qué esquema es el más adecuado?
- (a) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.  
(b) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.  
(c) Vuelta atrás, es el esquema más eficiente para este problema.
38. 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) Divide y vencerás.  
(b) Programación dinámica.  
(c) Ramificación y poda.
39. La eficiencia de los algoritmos voraces se basa en...
- (a) ... el hecho de que, con antelación, las posibles decisiones se ordenan de mejor a peor.  
(b) ... el hecho de que las decisiones tomadas no se reconsidernan.  
(c) En el esquema voraz no se puede hablar de eficiencia puesto que a menudo no resuelve el problema.

40. 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.

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

**[A] o [C]**

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

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

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

10. ¿Cuál de estas tres expresiones es cierta?

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

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

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

12. Indicad cuál de estas tres expresiones es falsa:

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

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

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

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

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

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

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

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

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

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

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

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

19. Las relaciones de recurrencia...

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

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

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

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

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

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

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

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

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

NO SEGURO

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

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

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

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

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

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

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

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

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

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

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

29. ¿Cuál de estas tres expresiones es falsa?

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

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

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

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

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

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

Seleccione una:

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

¿Cuál de estas tres expresiones es falsa?

Seleccione una

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

Sin contestar  
Puntúa como 1,00  
▼ Marcar pregunta

$$f(n) = \begin{cases} \Theta(1) & n=0 \\ \Theta(1)+f(n/3) & n>0 \end{cases}$$

Seleccione una:

- a.  $f(n) \in \Theta(\log(n))$ .
- b.  $f(n) \in \Theta(n/3)$
- c. Ninguna de las otras dos es cierta.

La respuesta correcta es:  $f(n) \in \Theta(\log(n))$ .

[Finalizar revisión](#)

---

Contacto: [ite.moodle@ua.es](mailto:ite.moodle@ua.es)  
Tutorial Moodle UA

## 2013-14\_ANALISIS Y DISEÑO DE ALGORITMOS\_34018

Página Principal ► Mis cursos ► Ingeniería y Arquitectura ► ADA\_34018 ► Simulacro del primer examen parcial de ADA ► Entrenamiento\_primer\_parcial

|                                                                                                                |
|----------------------------------------------------------------------------------------------------------------|
| <b>Navegación por el cuestionario</b>                                                                          |
| <a href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a> <a href="#">5</a> <a href="#">6</a>    |
| <a href="#">7</a> <a href="#">8</a> <a href="#">9</a> <a href="#">10</a> <a href="#">11</a> <a href="#">12</a> |
| <a href="#">13</a> <a href="#">14</a> <a href="#">15</a>                                                       |
| <a href="#">Mostrar una página cada vez</a>                                                                    |
| <a href="#">Finalizar revisión</a>                                                                             |

|                        |                                        |
|------------------------|----------------------------------------|
| <b>Comenzado el</b>    | viernes, 7 de marzo de 2014, 17:46     |
| <b>Estado</b>          | Finalizado                             |
| <b>Finalizado en</b>   | viernes, 7 de marzo de 2014, 17:52     |
| <b>Tiempo empleado</b> | 5 minutos 47 segundos                  |
| <b>Puntos</b>          | 0,00/15,00                             |
| <b>Calificación</b>    | <b>0,00</b> de un máximo de 10,00 (0%) |

### Pregunta 1

Sin contestar

Puntúa como 1,00

 Marcar pregunta

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

Seleccione una:

- a. ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.
- b. Las demás opciones son verdaderas.
- c. ... es una función de la talla que tiene que estar definida para todos los posibles valores de ésta.

La respuesta correcta es: ... es una función de la talla que tiene que estar definida para todos los posibles valores de ésta.

### Pregunta 2

Sin contestar

Puntúa como 1,00

 Marcar pregunta

Sobre la complejidad temporal de la siguiente función:

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

Seleccione una:

- a. Ninguna de las otras dos alternativas es cierta.
- b. Las complejidades en los casos mejor y peor son distintas.
- c. El mejor de los casos se da cuando  $n \leq 1$  y en tal caso la complejidad es constante.

La respuesta correcta es: Ninguna de las otras dos alternativas es cierta.

### Pregunta 3

Sin contestar

Puntúa como 1,00

 Marcar pregunta

Con respecto al esquema *Divide y vencerás*, ¿es cierta la siguiente afirmación?

Si la talla se reparte equitativamente entre los subproblemas, entonces la complejidad temporal resultante es una función logarítmica.

Seleccione una:

- a. No, nunca, puesto que también hay que añadir el coste de la división en subproblemas y la posterior combinación.
- b. No tiene porqué, la complejidad temporal no depende únicamente del tamaño resultante de los subproblemas.
- c. Sí, siempre, en Divide y Vencerás la complejidad temporal depende únicamente del tamaño de los subproblemas.

La respuesta correcta es: No tiene porqué, la complejidad temporal no depende únicamente del tamaño resultante de los subproblemas.

**Pregunta 4**

Sin contestar

Puntúa como 1,00

Marcar pregunta

¿Qué cota se deduce de la siguiente relación de recurrencia?

$$f(n) = \begin{cases} 1 & n=1 \\ n+4f(n/2) & n>1 \end{cases}$$

Seleccione una:

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

La respuesta correcta es:  $f(n) \in \Theta(n^2)$ **Pregunta 5**

Sin contestar

Puntúa como 1,00

Marcar pregunta

¿Cuál de estas tres expresiones es falsa?

Seleccione una:

- a.  $2n^3 - 10n^2 + 1 \in O(n^3)$
- b.  $n + n\sqrt{n} \in \Omega(n)$
- c.  $n + n\sqrt{n} \in \Theta(n)$

La respuesta correcta es:  $n + n\sqrt{n} \in \Theta(n)$ **Pregunta 6**

Sin contestar

Puntúa como 1,00

Marcar pregunta

Sea  $f(n) = n \log(n) + n$ .

Seleccione una:

- a. ...  $f(n) \in \Omega(n \log(n))$
- b. ...  $f(n) \in O(n \log(n))$
- c. Las otras dos opciones son ciertas

La respuesta correcta es: Las otras dos opciones son ciertas

**Pregunta 7**

Sin contestar

Puntúa como 1,00

Marcar pregunta

Si  $f_1(n) \in O(g_1(n))$  y  $f_2(n) \in O(g_2(n))$  entonces...

Seleccione una:

- a. Las otras dos alternativas son ciertas.
- b.  $f_1(n) + f_2(n) \in O(\max(g_1(n), g_2(n)))$
- c.  $f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$

La respuesta correcta es: Las otras dos alternativas son ciertas.

**Pregunta 8**

Sin contestar

Puntúa como 1,00

Marcar pregunta

¿Cuál es la complejidad temporal de la siguiente función?

```
int ejemplo (vector<int> & v) {
 int n=v.size();
 int j,i=2;
 int sum=0;
 while (n>0 && i<n) {
 j=i;
 while (v[j] != v[1]) {
 sum+=v[j];
 j=j/2;
 }
 }
}
```

```

 i++;
 }
 return sum;
}

```

Seleccione una:

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

La respuesta correcta es:  $\Theta(n \log n)$

### Pregunta 9

Sin contestar

Puntúa como 1,00

 Marcar pregunta

En cuanto a la complejidad temporal de la siguiente función:

```

int ejemplo (vector < int > & v) {
 int n=v.size();
 int j,i=2;
 int sum=0;
 while (n>0 && i<n) {
 j=i;
 while (v[j] != v[1]){
 sum+=v[j];
 j=j/2;
 }
 i++;
 }
 return sum;
}

```

Seleccione una:

- a. Las complejidades en el mejor y en el peor de los casos no coinciden.
- b. El mejor de los casos se da cuando  $n = 0$ , su complejidad es constante.
- c. Esta función no presenta casos mejor y peor puesto que sólo puede haber una instancia para cada una de las posibles talla

La respuesta correcta es: Las complejidades en el mejor y en el peor de los casos no coinciden.

### Pregunta 10

Sin contestar

Puntúa como 1,00

 Marcar pregunta

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

```

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

```

Seleccione una:

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

La respuesta correcta es:  $O(\log^2(n))$

### Pregunta 11

Sin contestar

Puntúa como 1,00

 Marcar pregunta

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

```

a = 0;
for(int i = 0; i < n*n; i++)
 a += A[(i + j) % n];

```

Seleccione una:

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

La respuesta correcta es:  $O(n^2)$

**Pregunta 12**

Sin contestar

Puntúa como 1,00

 Marcar pregunta

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

Seleccione una:

- a. ... no presenta caso mejor y peor distintos para instancias del mismo tamaño.
- b. ... es más eficiente si el vector ya está ordenado.
- c. ... es la versión con mejor complejidad en el mejor de los casos.

La respuesta correcta es: ... no presenta caso mejor y peor distintos para instancias del mismo tamaño.

**Pregunta 13**

Sin contestar

Puntúa como 1,00

 Marcar pregunta

El siguiente fragmento del algoritmo de ordenación *Quicksort* reorganiza los elementos del vector para obtener una subsecuencia de elementos menores que el pivote y otra de mayores. Su complejidad temporal, con respecto al tamaño del vector  $v$ , que está delimitado por los valores  $pi$  y  $pf$ , es...

```
x = v[pi];
i = pi+1;
j = pf;
do {
 while(i<=pf && v[i] < x) i++;
 while(v[j] > x) j--;
 if(i <= j) {
 swap(v[i],v[j]);
 i++;
 j--;
 }
} while(i < j);
swap(v[pi],v[j]);
```

Nota: La función swap se realiza en tiempo constante.

Seleccione una:

- a. ... lineal en cualquier caso.
- b. ... cuadrática en el peor de los casos.
- c. ... lineal en el caso peor y constante en el caso mejor.

La respuesta correcta es: ... lineal en cualquier caso.

**Pregunta 14**

Sin contestar

Puntúa como 1,00

 Marcar pregunta

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

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

Seleccione una:

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

La respuesta correcta es:  $f(n) \in \Omega(2^n)$

**Pregunta 15**

¿Cuál es la solución a la siguiente relación de recurrencia?

Sin contestar  
Puntúa como 1,00  
▼ Marcar pregunta

$$f(n) = \begin{cases} \Theta(1) & n=0 \\ \Theta(1)+f(n/3) & n>0 \end{cases}$$

Seleccione una:

- a.  $f(n) \in \Theta(\log(n))$ .
- b.  $f(n) \in \Theta(n/3)$
- c. Ninguna de las otras dos es cierta.

La respuesta correcta es:  $f(n) \in \Theta(\log(n))$ .

[Finalizar revisión](#)

---

Contacto: [ite.moodle@ua.es](mailto:ite.moodle@ua.es)  
Tutorial Moodle UA

## Respuestas para la modalidad 5

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  $\tilde{O}(n \log n)$ .
- (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.

2. Sea la siguiente relación de recurrencia

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

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

- (a)  $g(n) = 1$
- (b)  $g(n) = \log n$
- (c)  $g(n) = n$

3. 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) Sí, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.
- (c) No, ya que en cualquier caso se deben explorar todas las soluciones factibles.

4. 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) Lineal.
- (b) Cuadrática.
- (c) Cúbica.

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

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

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

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

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.

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

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

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

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.
11. ¿Cuál de estos problemas tiene una solución eficiente utilizando *programación dinámica*?
- (a) La mochila discreta sin restricciones adicionales.
  - (b) El problema de la asignación de tareas.
  - (c) El problema del cambio (devolver una cantidad de dinero con el mínimo número de monedas).
12. De las siguientes expresiones, o bien dos son verdaderas y una es falsa, o bien dos son falsas y una es verdadera. Marca la que (en este sentido) es distinta a las otras dos.
- (a)  $\log(n^3) \notin \Theta(\log_3(n))$
  - (b)  $\Theta(\log^2(n)) = \Theta(\log^3(n))$
  - (c)  $\Theta(\log_2(n)) = \Theta(\log_3(n))$
13. Se quiere reducir la complejidad temporal de la siguiente función haciendo uso de programación dinámica. ¿Cuál sería la complejidad temporal resultante?
- ```
unsigned g( unsigned n, unsigned r) {
    if (r==0 || r==n)
        return 1;
    return g(n-1, r-1) + g(n-1, r);
}
```
- (a) Cuadrática
 - (b) Se puede reducir hasta lineal.
 - (c) La función no cumple con los requisitos necesarios para poder aplicar programación dinámica.
14. De las siguientes expresiones, o bien dos son verdaderas y una es falsa, o bien dos son falsas y una es verdadera. Marca la que (en este sentido) es distinta a las otras dos.
- (a) $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$
 - (b) $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$
 - (c) $(4^{\log_2(n)}) \subset O(n) \subset O(2^n)$

15. Dada la relación de recurrencia:

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

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

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

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

- (a) El problema de la mochila discreta.
- (b) El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
- (c) El problema de la asignación de coste mínimo de n tareas a n trabajadores cuando el coste de asignar la tarea i al trabajador j , c_{ij} está tabulado en una matriz.

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

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

18. 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) ... pueden eliminar soluciones parciales que son factibles.
- (c) Las otras dos opciones son ambas son verdaderas.

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

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

20. 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 convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.
 - (c) Se puede mejorar la eficiencia del algoritmo definiendo de antemano el orden en el que se deben calcular las soluciones a los subproblemas y llenando una tabla en ese orden.
21. En el problema del coloreado de grafos (mínimo número de colores necesarios para colorear todos los vértices de un grafo de manera que no queden dos adyacentes con el mismo color) resuelto mediante *ramificación y poda*, ¿De qué manera se debería ordenar la lista de nodos vivos para obtener una solución aceptable?
- (a) las otras dos opciones pueden ser ambas correctas.
 - (b) Por cota optimista: explorando primero los nodos con menor cota optimista.
 - (c) Por cota pesimista: explorando primero los nodos con menor cota pesimista.
22. Si $\lim_{n \rightarrow \infty} (f(n)/n^2) = k$, y $k \neq 0$, ¿cuál de estas tres afirmaciones es cierta?
- (a) $f(n) \in \Omega(n^3)$
 - (b) $f(n) \in \Theta(n^2)$
 - (c) $f(n) \in \Theta(n^3)$
23. 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) Sí, puesto que es posible que a pesar de utilizar dichas cotas no se descarte ningún nodo.
 - (c) No, las cotas tanto optimistas como pesimistas garantizan la reducción del espacio de soluciones y por tanto la eficiencia del algoritmo.
24. Si $f \in \Theta(g_1)$ y $f \in \Theta(g_2)$ entonces
- (a) $f \in \Theta(g_1 \cdot g_2)$
 - (b) $f \notin \Theta(\max(g_1, g_2))$
 - (c) $f \in \Theta(g_1 + g_2)$

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

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

- (a) ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- (b) ... un recorrido guiado por estimaciones de las que pueden ser las mejores ramas del árbol que representa el espacio de soluciones.
- (c) ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.

28. 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.

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

- (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.

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

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

Di cuál de las siguientes afirmaciones es *cierta*:

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

31. En un algoritmo de *ramificación y poda*, el orden escogido para priorizar los nodos en la lista de nodos vivos ...

- (a) ... determina la complejidad temporal en el peor de los casos del algoritmo.
- (b) ... nunca afecta al tiempo necesario para encontrar la solución óptima.
- (c) ... puede influir en el número de nodos que se descartan sin llegar a expandirlos.

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.

33. ¿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 actualizar el valor de la mejor solución hasta el momento.
- (c) Para obtener una cota optimista más precisa.

34. 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.

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.

Respuestas para la modalidad 1

1. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...
 - (a) ... pueden eliminar vectores que representan posibles soluciones factibles.
 - (b) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.
 - (c) Las otras dos opciones son ambas verdaderas.
2. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?
 - (a) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
 - (b) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
 - (c) Vuelta atrás, es el esquema más eficiente para este problema.
3. 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) $n + n \log n \in \Omega(n + n \log n)$
 - (b) $O(n^2) \subset O(2^{\log n})$
 - (c) $\Omega(n^2) \subset \Omega(n)$
4. Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Si quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?
 - (a) un vector de booleanos.
 - (b) un par de enteros que indiquen los cortes realizados y el valor acumulado.
 - (c) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.
5. El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continúe ordenado es...
 - (a) ... $O(n)$.
 - (b) ... $O(\log n)$.
 - (c) ... $\Omega(n^2)$.

6. Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:
- (a) El valor de la mochila continua correspondiente .
 - (b) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
 - (c) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
7. ¿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) El coste temporal promedio.
 - (b) El coste temporal asintótico en el caso medio.
 - (c) Nada de interés.
8. ¿Qué estrategia de búsqueda es a priori más apropiada en un esquema de vuelta atrás?
- (a) En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.
 - (b) Explorar primero los nodos con mejor cota optimista.
 - (c) Explorar primero los nodos que están más completados.
9. ¿Qué ocurre si la cota optimista de un nodo resulta ser el valor que se obtiene de una solución factible pero que no es la mejor del subárbol generado por ese nodo?
- (a) Nada especial, las cotas optimistas se corresponden con soluciones factibles que no tienen porqué ser las mejores.
 - (b) Que el algoritmo sería incorrecto pues podría descartarse el nodo que conduce a la solución óptima.
 - (c) Que el algoritmo sería más lento pues se explorarían más nodos de los necesarios.
10. 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) De una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización.
 - (b) De una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización.
 - (c) 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.

11. Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 8T\left(\frac{n}{8}\right) + g(n) & \text{en otro caso} \end{cases}$$

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

- (a) $g(n) = n^2$
- (b) $g(n) = n^3$
- (c) $g(n) = n$

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

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

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

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

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

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

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

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

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

- (a) Sí, en caso de existir con este esquema siempre se puede encontrar un camino de salida
- (b) No, para garantizar que se encuentra un camino de salida hay que aplicar métodos de búsqueda exhaustiva como vuelta atrás o ramificación y poda.
- (c) 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.

17. ¿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) $\Theta(n^2)$
- (b) $\Theta(n \times m)$
- (c) $\Theta(n)$

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

- (a) ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a .
- (b) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.
- (c) Las otras dos opciones son ambas verdaderas.

19. Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?

- (a) Que se podría podar el nodo que conduce a la solución óptima.
- (b) Que se podría explorar más nodos de los necesarios.
- (c) Que se podría explorar menos nodos de los necesarios.

20. ¿Qué complejidad se obtiene a partir de la relación de recurrencia $T(n) = 9T(n/3) + n^3$ con $T(1) = O(1)$?

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

21. Si $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$ entonces ...
- (a) $\dots f(n) \in O(g(n))$
 (b) $\dots f(n) \in \Omega(g(n))$
(c) $\dots f(n) \in \Theta(g(n))$
22. Se desea ordenar una lista enlazada de n elementos haciendo uso del algoritmo *Mergesort*. En este caso, al tratarse de una lista, la complejidad temporal asintótica de realizar la división en subproblemas resulta ser lineal con el tamaño de esa lista. ¿Cuál sería entonces el coste temporal de realizar dicha ordenación?
- (a) $\Theta(n^2)$
(b) Ninguna de las otras dos opciones es cierta.
 (c) $\Theta(n \log n)$
23. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
- (a) 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.
 (b) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
(c) Lo más importante es conseguir una cota pesimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.

24. 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 en el mejor de los casos es $\Omega(1)$
 - (b) La complejidad temporal en el mejor de los casos es $\Omega(n)$
 - (c) La complejidad temporal exacta es $\Theta(n \log n)$
25. En el esquema de ramificación y poda, ¿qué estructura es la más adecuada si queremos realizar una exploración por niveles?
- (a) Pila
 - (b) Cola
 - (c) Cola de prioridad

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

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

Di cuál de las siguientes afirmaciones es cierta:

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

27. 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) Ninguna de las otras dos recurrencias se corresponde con un esquema de divide y vencerás.
- (c) $nc(n, m) = nc(n - 1, m) + nc(n, m - 1) + nc(n - 1, m - 1)$

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

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

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

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

- (a) $\Omega(n^2) \subset \Omega(n^3)$
- (b) $O(n^2) \subset O(n^3)$
- (c) $\Theta(n^2) \subset \Theta(n^3)$

31. Dado el problema del laberinto con tres movimientos, ¿cuál de las estrategias siguientes proveería de una cota optimista para ramificación y poda?
- (a) Las otras dos estrategias son ambas válidas.
(b) Suponer que en adelante todas las casillas del laberinto son accesibles.
(c) Suponer que ya no se van a realizar más movimientos.
32. Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones recursiva expresa mejor su complejidad temporal para el caso general, siendo n el número de discos?
- (a) $T(n) = 2T(n - 1) + n$
(b) $T(n) = T(n - 1) + n$
(c) $T(n) = 2T(n - 1) + 1$
33. Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es par y que se utilizará la siguiente recursividad: $\text{pot}(x, n) = \text{pot}(x, n/2) * \text{pot}(x, n/2)$; ¿Qué esquema resulta ser más eficiente en cuanto al coste temporal?
- (a) Divide y vencerás.
(b) En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.
(c) Programación dinámica.
34. ¿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.

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

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

Marcad la opción correcta.

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

36. El esquema de vuelta atrás ...

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

37. Si $f \notin O(g_1)$ y $f \in O(g_2)$ entonces siempre se cumplirá:

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

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

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

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

40. ¿Qué tienen en común los algoritmos de ordenación *Quicksort* y *Mergesort*.

- (a) La complejidad temporal de la división en subproblemas.
- (b) La complejidad temporal de la combinación de las soluciones parciales.
- (c) El número de llamadas recursivas que hacen en el mejor de los casos.

Respuestas para la modalidad 8

1. 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) Si, siempre es así.
 - (b) En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
 - (c) En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
2. ¿Cuál de estos problemas tiene una solución eficiente utilizando *programación dinámica*?
 - (a) El problema de la asignación de tareas.
 - (b) La mochila discreta sin restricciones adicionales.
 - (c) El problema del cambio.
3. 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) Sólo si $d \log d > k n$ (donde k es una constante que depende de la implementación)
 - (b) Sí, ya que el *mergesort* es $O(n \log n)$ y este es $O(n)$
 - (c) No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.
4. En un algoritmo de *ramificación y poda*, si la lista de nodos vivos no está ordenada de forma apropiada ...
 - (a) ... podría ocurrir que se descarten nodos factibles.
 - (b) ... podría ocurrir que se pade el nodo que conduce a la solución óptima.
 - (c) ... podría ocurrir que se exploren nodos de forma innecesaria.
5. El uso de funciones de cota en ramificación y poda ...
 - (a) ... transforma en polinómicas complejidades que antes eran exponenciales.
 - (b) ... puede reducir el número de instancias del problema que pertenecen al caso peor.
 - (c) ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.

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

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

Di cuál de las siguientes afirmaciones es falsa:

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

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

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

8. 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 convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.
- (b) Se puede mejorar la eficiencia del algoritmo guardando en una tabla el valor devuelto para cada conjunto de parámetros de cada llamada cuando ésta se produce por primera vez.
- (c) Se puede mejorar la eficiencia del algoritmo definiendo de antemano el orden en el que se deben calcular las soluciones a los subproblemas y llenando una tabla en ese orden.

9. 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) cuadrática
- (b) cúbica
- (c) exponencial

10. ¿Para cuál de estos problemas de optimización se conoce una solución voraz?

- (a) El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
- (b) El problema de la mochila discreta.
- (c) El problema de la asignación de coste mínimo de n tareas a n trabajadores cuando el coste de asignar la tarea i al trabajador j , c_{ij} está tabulado en una matriz.

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

12. 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 **XXXXXX**?

```

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(XXXXXX) );
    }
    m[n]=q;
    return q;
}

```

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

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

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

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

- (a) No
- (b) Sí, en cualquier caso.
- (c) Sí, pero siempre que la complejidad temporal conjunta de las operaciones de descomposición del problema y la combinación de las soluciones sea polinómica.

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

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

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

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

18. En los algoritmos de *ramificación y poda* ...

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

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

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

20. 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.
21. Cuando se resuelve el problema de la mochila discreta usando la estrategia de vuelta atrás, ¿puede ocurrir que se tarde menos en encontrar la solución óptima si se prueba primero a meter cada objeto antes de no meterlo?
- (a) Sí, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.
 - (b) Sí, tanto si se usan cotas optimistas para podar el árbol de búsqueda como si no.
 - (c) No, ya que en cualquier caso se deben explorar todas las soluciones factibles.
22. Estudiad la relación de recurrencia:
- $$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT\left(\frac{n}{q}\right) + g(n) & \text{en otro caso} \end{cases}$$
- (donde p y q son enteros mayores que 1). Di cuál de los siguientes esquemas algorítmicos produce de manera natural relaciones de recurrencia así.
- (a) Ramificación y poda
 - (b) Programación dinámica
 - (c) Divide y vencerás
23. ¿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) La estrategia voraz
 - (b) Para este problema, las dos estrategias citadas serían similares en cuanto a eficiencia
 - (c) Programación dinámica

24. 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) Ω(n) y O(n2)
(b) Ω(n) y O(2n)
(c) Θ(2n)
```

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

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

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

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

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

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

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

29. Sea la siguiente relación de recurrencia

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

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

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

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

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

- (a) ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.
- (b) ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- (c) ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.

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

33. 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) Calcular la distancia recorrida moviéndose al azar por el grafo hasta llegar (por azar) a la ciudad destino.
 - (b) Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.
 - (c) Utilizar la solución (subóptima) que se obtiene al resolver el problema mediante un algoritmo voraz.
34. Uno de estos tres problemas no tiene una solución trivial y eficiente que siga el esquema voraz.
- (a) El problema de la mochila discreta sin limitación en la carga máxima de la mochila.
 - (b) El problema de la mochila continua.
 - (c) El problema del cambio.
35. 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 dos anteriores son ciertas.
36. Si un problema de optimización lo es para una función que toma valores continuos ...
- (a) La programación dinámica recursiva puede resultar mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.
 - (b) La programación dinámica iterativa siempre es mucho más eficiente que la programación dinámica recursiva 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.
37. Si $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$ entonces ...
- (a) ... $f(n) \in O(g(n))$
 - (b) ... $g(n) \in O(f(n))$
 - (c) ... $f(n) \in \Theta(g(n))$

38. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

- (a) ... pueden eliminar soluciones parciales que son factibles.
- (b) ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- (c) Las dos anteriores son verdaderas.

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

40. 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.pessimistic_b() <= pb
- (c) n.optimistic_b() >= pb

Respuestas para la modalidad 1

1. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1, 1)$ hasta la casilla (n, m) y para ello se aplica el esquema programación dinámica para obtener un algoritmo lo más eficiente posible en cuanto a complejidad temporal y espacial. ¿Cuáles serían ambas complejidades?
 - (a) Temporal $\Theta(n \times m)$ y espacial $\Theta(\min\{n, m\})$
 - (b) Temporal $\Theta(\max\{n, m\})$ y espacial $\Theta(\max\{n, m\})$
 - (c) Temporal $\Theta(n \times m)$ y espacial $\Theta(n \times m)$
2. 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) De una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización.
 - (b) De una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización.
 - (c) 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.
3. ¿Qué se deduce de $f(n)$ y $g(n)$ si se cumple $\lim_{n \rightarrow \infty} (f(n)/g(n)) = k$, con $k \neq 0$?
 - (a) $f(n) \in O(g(n))$ pero $g(n) \notin O(f(n))$
 - (b) $g(n) \in O(f(n))$ pero $f(n) \notin O(g(n))$
 - (c) $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
4. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?
 - (a) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
 - (b) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
 - (c) Vuelta atrás, es el esquema más eficiente para este problema.

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

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

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

7. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

- (a) ... pueden eliminar vectores que representan posibles soluciones factibles.
- (b) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.
- (c) Las otras dos opciones son ambas verdaderas.

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

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

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

10. Una de las prácticas de laboratorio consistió en el cálculo empírico de la complejidad temporal promedio del algoritmo de ordenación de vectores Quicksort tomando como centinela el elemento del vector que ocupa la posición central. ¿Cuál es el orden de complejidad que se obtuvo?

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

11. Sea la siguiente relación de recurrencia

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

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

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

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

13. ¿Qué estrategia de búsqueda es a priori más apropiada en un esquema de vuelta atrás?
- (a) En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.
 - (b) Explorar primero los nodos con mejor cota optimista.
 - (c) Explorar primero los nodos que están más completados.
14. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...
- (a) ... no se puede usar para resolver problemas de optimización.
 - (b) ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.
 - (c) ... debe recorrer siempre todo el árbol.
15. Un algoritmo recursivo basado en el esquema *divide y vencerás*...
- (a) ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a .
 - (b) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.
 - (c) Las otras dos opciones son ambas verdaderas.
16. 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 en el mejor de los casos es $\Omega(n)$
- (b) La complejidad temporal en el mejor de los casos es $\Omega(1)$
- (c) La complejidad temporal exacta es $\Theta(n^2)$

17. Se desea ordenar una lista enlazada de n elementos haciendo uso del algoritmo *Mergesort*. En este caso, al tratarse de una lista, la complejidad temporal asintótica de realizar la división en subproblemas resulta ser lineal con el tamaño de esa lista. ¿Cuál sería entonces el coste temporal de realizar dicha ordenación?
- (a) $\Theta(n^2)$
 - (b) Ninguna de las otras dos opciones es cierta.
 - (c) $\Theta(n \log n)$
18. Dado el problema del laberinto con tres movimientos, ¿se puede aplicar un esquema de programación dinámica para obtener un camino de salida?
- (a) Sí, en caso de existir con este esquema siempre se puede encontrar un camino de salida
 - (b) No, para garantizar que se encuentra un camino de salida hay que aplicar métodos de búsqueda exhaustiva como vuelta atrás o ramificación y poda.
 - (c) 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.
19. Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿qué propiedad cumple una cota optimista?
- (a) Siempre es mayor o igual que la mejor solución posible alcanzada.
 - (b) Las otras dos opciones son ambas falsas.
 - (c) Asegura un ahorro en la comprobación de todas las soluciones factibles.
20. De las siguientes expresiones, o bien dos son verdaderas y una es falsa, o bien dos son falsas y una es verdadera. Marca la que (en este sentido) es distinta a las otras dos.
- (a) $n + n \log n \in \Omega(n)$
 - (b) $O(2^{\log n}) \subset O(n^2)$
 - (c) $\Theta(n) \subset \Theta(n^2)$
21. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
- (a) 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.
 - (b) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
 - (c) Lo más importante es conseguir una cota pesimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.

22. ¿Qué complejidad se obtiene a partir de la relación de recurrencia $T(n) = 8T(n/2) + n^3$ con $T(1) = O(1)$?
- (a) $O(n^3)$
 - (b) $O(n \log n)$
 - (c) $O(n^3 \log n)$
23. 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) Divide y vencerás.
 - (b) Programación dinámica.
 - (c) Ramificación y poda.
24. ¿En ramificación y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?
- (a) No, la cota optimista sólo se utiliza para determinar si una *n-tupla* es prometedora.
 - (b) Sí, aunque no es una garantía de que sea una buena estrategia de búsqueda.
 - (c) Sí, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista.
25. El esquema voraz ...
- (a) Garantiza encontrar una solución a cualquier problema, aunque puede que no sea óptima.
 - (b) Puede que no encuentre una solución pero si lo hace se garantiza que es óptima.
 - (c) Las otras dos opciones son ambas falsas.
26. Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:
- (a) El valor de la mochila continua correspondiente .
 - (b) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
 - (c) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
27. 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) Ninguna de las otras dos recurrencias se corresponde con un esquema de divide y vencerás.
 - (c) $nc(n, m) = nc(n - 1, m) + nc(n, m - 1) + nc(n - 1, m - 1)$

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

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

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

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

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

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

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

Marcad la opción correcta.

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

32. ¿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) $\Theta(n^2)$
- (b) $\Theta(nm)$
- (c) $\Theta(n)$

33. Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones recursiva expresa mejor su complejidad temporal para el caso general, siendo n el número de discos?

- (a) $T(n) = 2T(n - 1) + n$
- (b) $T(n) = T(n - 1) + n$
- (c) $T(n) = 2T(n - 1) + 1$

34. Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?

- (a) un vector de booleanos.
- (b) un par de enteros que indiquen los cortes realizados y el valor acumulado.
- (c) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.

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

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

36. Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es par y que se utilizará la siguiente recurrencia: $\text{pot}(x, n) = \text{pot}(x, n/2) * \text{pot}(x, n/2)$; ¿Qué esquema resulta ser más eficiente en cuanto al coste temporal?
- (a) Divide y vencerás.
(b) En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.
 (c) Programación dinámica.
37. Si $f \in \Omega(g_1)$ y $f \in \Omega(g_2)$ entonces
- (a) $f \notin \Omega(\min(g_1, g_2))$
(b) $f \in \Omega(g_1 \cdot g_2)$
 (c) $f \in \Omega(g_1 + g_2)$
38. El esquema de vuelta atrás ...
- (a) Las otras dos opciones son ambas verdaderas.
 (b) Garantiza que encuentra la solución óptima a cualquier problema de selección discreta.
(c) Se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado.
39. Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?
- (a) Que se podría podar el nodo que conduce a la solución óptima.
 (b) Que se podría explorar más nodos de los necesarios.
(c) Que se podría explorar menos nodos de los necesarios.
40. El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continue ordenado es...
- (a) ... $O(n)$.
(b) ... $O(\log n)$.
(c) ... $\Omega(n^2)$.

Respuestas para la modalidad F

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

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

2. 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 $O(n^2)$.

3. 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

4. Sea la siguiente relación de recurrencia

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

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

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

5. ¿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.

6. 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.

7. 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) $O(n^2)$

8. 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.

9. Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?
- (a) Se multiplica k por la distancia de la arista más corta que nos queda por considerar, donde k es el número de saltos que nos quedan por dar.
 - (b) Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
 - (c) Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las k aristas más cortas, donde k es el número de saltos que nos quedan por dar.
10. 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 recursiva 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.
11. 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.
12. 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 $O(n^2) \subset O(n^3)$
13. 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.

14. 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.
15. 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.
16. 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.

17. 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(XXXXXXXX));  
    }  
    r[n]=q;  
    return q;  
}
```

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

18. 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.

19. 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.

20. 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.
21. 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.
22. 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
23. 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.
24. 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 $f(n) \in O(n^2)$

25. 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.
26. 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.
27. 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.
28. 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.
29. 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.

30. 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) $O(\sqrt{n})$
31. Se desea encontrar el camino más corto entre dos ciudades. Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geométrica (en línea recta) entre cada par de ciudades. Se pretende acelerar la búsqueda de un algoritmo de *ramificación y poda* priorizando los nodos vivos (ciudades) que estén a menor distancia geográfica de la ciudad objetivo.
- (a) El nuevo algoritmo no garantiza que vaya a ser más rápido para todas las instancias del problema posibles.
 - (b) El nuevo algoritmo siempre será más rápido.
 - (c) Esta estrategia no asegura que se obtenga el camino más corto.
32. 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
33. 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.
34. ¿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.

35. ¿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.
36. Cuando se usa un algoritmo voraz para abordar la resolución de un problema de optimización por selección discreta (es decir, un problema para el cual la solución consiste en encontrar un subconjunto del conjunto de elementos que optimiza una determinada función), ¿cuál de estas tres cosas es imposible que ocurra?
- (a) Que el algoritmo no encuentre ninguna solución.
 - (b) Que se reconsidera la decisión ya tomada anteriormente respecto a la selección de un elemento a la vista de la decisión que se debe tomar en el instante actual.
 - (c) Que la solución no sea la óptima.
37. 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.
38. 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.
39. 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.

40. 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.

Respuestas para la modalidad 1

1. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1, 1)$ hasta la casilla (n, m) y para ello se aplica el esquema programación dinámica para obtener un algoritmo lo más eficiente posible en cuanto a complejidad temporal y espacial. ¿Cuáles serían ambas complejidades?
 - (a) Temporal $\Theta(n \times m)$ y espacial $\Theta(\min\{n, m\})$
 - (b) Temporal $\Theta(\max\{n, m\})$ y espacial $\Theta(\max\{n, m\})$
 - (c) Temporal $\Theta(n \times m)$ y espacial $\Theta(n \times m)$
2. 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) De una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización.
 - (b) De una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización.
 - (c) 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.
3. ¿Qué se deduce de $f(n)$ y $g(n)$ si se cumple $\lim_{n \rightarrow \infty} (f(n)/g(n)) = k$, con $k \neq 0$?
 - (a) $f(n) \in O(g(n))$ pero $g(n) \notin O(f(n))$
 - (b) $g(n) \in O(f(n))$ pero $f(n) \notin O(g(n))$
 - (c) $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
4. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?
 - (a) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
 - (b) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
 - (c) Vuelta atrás, es el esquema más eficiente para este problema.

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

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

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

7. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

- (a) ... pueden eliminar vectores que representan posibles soluciones factibles.
- (b) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.
- (c) Las otras dos opciones son ambas verdaderas.

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

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

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

10. Una de las prácticas de laboratorio consistió en el cálculo empírico de la complejidad temporal promedio del algoritmo de ordenación de vectores Quicksort tomando como centinela el elemento del vector que ocupa la posición central. ¿Cuál es el orden de complejidad que se obtuvo?

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

11. Sea la siguiente relación de recurrencia

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

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

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

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

13. ¿Qué estrategia de búsqueda es a priori más apropiada en un esquema de vuelta atrás?
- (a) En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.
 - (b) Explorar primero los nodos con mejor cota optimista.
 - (c) Explorar primero los nodos que están más completados.
14. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...
- (a) ... no se puede usar para resolver problemas de optimización.
 - (b) ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.
 - (c) ... debe recorrer siempre todo el árbol.
15. Un algoritmo recursivo basado en el esquema *divide y vencerás*...
- (a) ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a .
 - (b) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.
 - (c) Las otras dos opciones son ambas verdaderas.
16. 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 en el mejor de los casos es $\Omega(n)$
- (b) La complejidad temporal en el mejor de los casos es $\Omega(1)$
- (c) La complejidad temporal exacta es $\Theta(n^2)$

17. Se desea ordenar una lista enlazada de n elementos haciendo uso del algoritmo *Mergesort*. En este caso, al tratarse de una lista, la complejidad temporal asintótica de realizar la división en subproblemas resulta ser lineal con el tamaño de esa lista. ¿Cuál sería entonces el coste temporal de realizar dicha ordenación?
- (a) $\Theta(n^2)$
 - (b) Ninguna de las otras dos opciones es cierta.
 - (c) $\Theta(n \log n)$
18. Dado el problema del laberinto con tres movimientos, ¿se puede aplicar un esquema de programación dinámica para obtener un camino de salida?
- (a) Sí, en caso de existir con este esquema siempre se puede encontrar un camino de salida
 - (b) No, para garantizar que se encuentra un camino de salida hay que aplicar métodos de búsqueda exhaustiva como vuelta atrás o ramificación y poda.
 - (c) 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.
19. Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿qué propiedad cumple una cota optimista?
- (a) Siempre es mayor o igual que la mejor solución posible alcanzada.
 - (b) Las otras dos opciones son ambas falsas.
 - (c) Asegura un ahorro en la comprobación de todas las soluciones factibles.
20. De las siguientes expresiones, o bien dos son verdaderas y una es falsa, o bien dos son falsas y una es verdadera. Marca la que (en este sentido) es distinta a las otras dos.
- (a) $n + n \log n \in \Omega(n)$
 - (b) $O(2^{\log n}) \subset O(n^2)$
 - (c) $\Theta(n) \subset \Theta(n^2)$
21. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
- (a) 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.
 - (b) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
 - (c) Lo más importante es conseguir una cota pesimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.

22. ¿Qué complejidad se obtiene a partir de la relación de recurrencia $T(n) = 8T(n/2) + n^3$ con $T(1) = O(1)$?
- (a) $O(n^3)$
 - (b) $O(n \log n)$
 - (c) $O(n^3 \log n)$
23. 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) Divide y vencerás.
 - (b) Programación dinámica.
 - (c) Ramificación y poda.
24. ¿En ramificación y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?
- (a) No, la cota optimista sólo se utiliza para determinar si una *n-tupla* es prometedora.
 - (b) Sí, aunque no es una garantía de que sea una buena estrategia de búsqueda.
 - (c) Sí, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista.
25. El esquema voraz ...
- (a) Garantiza encontrar una solución a cualquier problema, aunque puede que no sea óptima.
 - (b) Puede que no encuentre una solución pero si lo hace se garantiza que es óptima.
 - (c) Las otras dos opciones son ambas falsas.
26. Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:
- (a) El valor de la mochila continua correspondiente .
 - (b) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
 - (c) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
27. 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) Ninguna de las otras dos recurrencias se corresponde con un esquema de divide y vencerás.
 - (c) $nc(n, m) = nc(n - 1, m) + nc(n, m - 1) + nc(n - 1, m - 1)$

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

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

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

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

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

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

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

Marcad la opción correcta.

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

32. ¿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) $\Theta(n^2)$
- (b) $\Theta(nm)$
- (c) $\Theta(n)$

33. Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones recursiva expresa mejor su complejidad temporal para el caso general, siendo n el número de discos?

- (a) $T(n) = 2T(n - 1) + n$
- (b) $T(n) = T(n - 1) + n$
- (c) $T(n) = 2T(n - 1) + 1$

34. Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?

- (a) un vector de booleanos.
- (b) un par de enteros que indiquen los cortes realizados y el valor acumulado.
- (c) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.

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

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

36. Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es par y que se utilizará la siguiente recurrencia: $\text{pot}(x, n) = \text{pot}(x, n/2) * \text{pot}(x, n/2)$; ¿Qué esquema resulta ser más eficiente en cuanto al coste temporal?
- (a) Divide y vencerás.
(b) En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.
 (c) Programación dinámica.
37. Si $f \in \Omega(g_1)$ y $f \in \Omega(g_2)$ entonces
- (a) $f \notin \Omega(\min(g_1, g_2))$
(b) $f \in \Omega(g_1 \cdot g_2)$
 (c) $f \in \Omega(g_1 + g_2)$
38. El esquema de vuelta atrás ...
- (a) Las otras dos opciones son ambas verdaderas.
 (b) Garantiza que encuentra la solución óptima a cualquier problema de selección discreta.
(c) Se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado.
39. Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?
- (a) Que se podría podar el nodo que conduce a la solución óptima.
 (b) Que se podría explorar más nodos de los necesarios.
(c) Que se podría explorar menos nodos de los necesarios.
40. El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continue ordenado es...
- (a) ... $O(n)$.
(b) ... $O(\log n)$.
(c) ... $\Omega(n^2)$.

Respuestas para la modalidad 2

1. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...
 - (a) ... no se puede usar para resolver problemas de optimización.
 - (b) ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.
 - (c) ... debe recorrer siempre todo el árbol.
2. El esquema voraz ...
 - (a) Garantiza encontrar una solución a cualquier problema, aunque puede que no sea óptima.
 - (b) Puede que no encuentre una solución pero si lo hace se garantiza que es óptima.
 - (c) Las otras dos opciones son ambas falsas.
3. Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?
 - (a) Que se podría podar el nodo que conduce a la solución óptima.
 - (b) Que se podría explorar más nodos de los necesarios.
 - (c) Que se podría explorar menos nodos de los necesarios.
4. ¿Qué se deduce de $f(n)$ y $g(n)$ si se cumple $\lim_{n \rightarrow \infty} (f(n)/g(n)) = k$, con $k \neq 0$?
 - (a) $f(n) \in O(g(n))$ pero $g(n) \notin O(f(n))$
 - (b) $g(n) \in O(f(n))$ pero $f(n) \notin O(g(n))$
 - (c) $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
5. ¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?
 - (a) Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.
 - (b) Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.
 - (c) Que el algoritmo sería más lento pues se explorarían más nodos de los necesarios.
6. Un algoritmo recursivo basado en el esquema *divide y vencerás...*
 - (a) ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a .
 - (b) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.
 - (c) Las otras dos opciones son ambas verdaderas.

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

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

Marcad la opción correcta.

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

8. 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) Ninguna de las otras dos recurrencias se corresponde con un esquema de divide y vencerás.
- (c) $nc(n, m) = nc(n - 1, m) + nc(n, m - 1) + nc(n - 1, m - 1)$

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

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

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

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

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

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

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

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

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

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

15. Si el coste temporal de un algoritmo es $T(n)$, ¿cuál de las siguientes situaciones es imposible?
- (a) $T(n) \in O(n)$ y $T(n) \in \Theta(n)$
 - (b) $T(n) \in \Omega(n)$ y $T(n) \in \Theta(n^2)$
 - (c) $T(n) \in \Theta(n)$ y $T(n) \in \Omega(n^2)$
16. ¿Qué complejidad se obtiene a partir de la relación de recurrencia $T(n) = 8T(n/2) + n^3$ con $T(1) = O(1)$?
- (a) $O(n^3)$
 - (b) $O(n \log n)$
 - (c) $O(n^3 \log n)$
17. Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:
- (a) El valor de la mochila continua correspondiente .
 - (b) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
 - (c) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
18. ¿Qué estrategia de búsqueda es a priori más apropiada en un esquema de vuelta atrás?
- (a) En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.
 - (b) Explorar primero los nodos con mejor cota optimista.
 - (c) Explorar primero los nodos que están más completados.
19. ¿En ramificación y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?
- (a) No, la cota optimista sólo se utiliza para determinar si una *n-tupla* es prometedora.
 - (b) Sí, aunque no es una garantía de que sea una buena estrategia de búsqueda.
 - (c) Sí, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista.
20. El esquema de vuelta atrás ...
- (a) Las otras dos opciones son ambas verdaderas.
 - (b) Garantiza que encuentra la solución óptima a cualquier problema de selección discreta.
 - (c) Se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado.

21. 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) Divide y vencerás.
 (b) Programación dinámica.
(c) Ramificación y poda.
22. De las siguientes afirmaciones marca la que es verdadera.
- (a) Las cotas pesimistas no son compatibles con un esquema de vuelta atrás.
 (b) En un esquema de vuelta atrás, las cotas pesimistas no tienen sentido si lo que se pretende es obtener todas las soluciones factibles.
(c) El esquema de vuelta atrás no es compatible con el uso conjunto de cotas pesimistas y optimistas.
23. Una de las prácticas de laboratorio consistió en el cálculo empírico de la complejidad temporal promedio del algoritmo de ordenación de vectores Quicksort tomando como centinela el elemento del vector que ocupa la posición central. ¿Cuál es el orden de complejidad que se obtuvo?
- (a) n^2
 (b) $n \log n$
(c) $n \log^2 n$
24. Dado el problema del laberinto con tres movimientos, ¿cuál de las estrategias siguientes proveería de una cota optimista para ramificación y poda?
- (a) Las otras dos estrategias son ambas válidas.
(b) Suponer que en adelante todas las casillas del laberinto son accesibles.
(c) Suponer que ya no se van a realizar más movimientos.
25. Dado el problema del laberinto con tres movimientos, ¿se puede aplicar un esquema de programación dinámica para obtener un camino de salida?
- (a) Sí, en caso de existir con este esquema siempre se puede encontrar un camino de salida
(b) No, para garantizar que se encuentra un camino de salida hay que aplicar métodos de búsqueda exhaustiva como vuelta atrás o ramificación y poda.
(c) 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.

26. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.

- (a) 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.
- (b) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
- (c) Lo más importante es conseguir una cota pesimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.

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

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

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

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

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

- (a) $n + n \log n \in \Omega(n)$
- (b) $O(2^{\log n}) \subset O(n^2)$
- (c) $\Theta(n) \subset \Theta(n^2)$

31. ¿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) $\Theta(n^2)$
- (b) $\Theta(nm)$
- (c) $\Theta(n)$

32. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?

- (a) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
- (b) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
- (c) Vuelta atrás, es el esquema más eficiente para este problema.

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

34. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

- (a) ... pueden eliminar vectores que representan posibles soluciones factibles.
- (b) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.
- (c) Las otras dos opciones son ambas verdaderas.

35. Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si $T(n) \in O(n)$, ¿en cuál de estos tres casos nos podemos encontrar?

- (a) $g(n) = \log n$
- (b) $g(n) = \sqrt{n}$
- (c) Las otras dos opciones son ambas ciertas.

36. Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones de recurrencia expresa mejor su complejidad temporal para el caso general, siendo n el número de discos?

- (a) $T(n) = 2T(n - 1) + n$
- (b) $T(n) = T(n - 1) + n$
- (c) $T(n) = 2T(n - 1) + 1$

37. Si $f \in \Omega(g_1)$ y $f \in \Omega(g_2)$ entonces

- (a) $f \notin \Omega(\min(g_1, g_2))$
- (b) $f \in \Omega(g_1 \cdot g_2)$
- (c) $f \in \Omega(g_1 + g_2)$

38. 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) De una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización.
(b) De una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización.
(c) 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.
39. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1, 1)$ hasta la casilla (n, m) y para ello se aplica el esquema programación dinámica para obtener un algoritmo lo más eficiente posible en cuanto a complejidad temporal y espacial. ¿Cuáles serían ambas complejidades?
- (a) Temporal $\Theta(n \times m)$ y espacial $\Theta(\min\{n, m\})$
(b) Temporal $\Theta(\max\{n, m\})$ y espacial $\Theta(\max\{n, m\})$
(c) Temporal $\Theta(n \times m)$ y espacial $\Theta(n \times m)$
40. Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?
- (a) un vector de booleanos.
(b) un par de enteros que indiquen los cortes realizados y el valor acumulado.
(c) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.

Respuestas para la modalidad 3

1. Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si $T(n) \in O(n)$, ¿en cuál de estos tres casos nos podemos encontrar?

- (a) $g(n) = \sqrt{n}$
- (b) Las otras dos opciones son ambas ciertas.
- (c) $g(n) = \log n$

2. ¿Qué se deduce de $f(n)$ y $g(n)$ si se cumple $\lim_{n \rightarrow \infty} (f(n)/g(n)) = k$, con $k \neq 0$?

- (a) $g(n) \in O(f(n))$ pero $f(n) \notin O(g(n))$
- (b) $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
- (c) $f(n) \in O(g(n))$ pero $g(n) \notin O(f(n))$

3. ¿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) $\Theta(n \times m)$
- (b) $\Theta(n)$
- (c) $\Theta(n^2)$

4. Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?

- (a) Que se podría explorar más nodos de los necesarios.
- (b) Que se podría explorar menos nodos de los necesarios.
- (c) Que se podría podar el nodo que conduce a la solución óptima.

5. Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es par y que se utilizará la siguiente recurrencia: $\text{pot}(x, n) = \text{pot}(x, n/2) * \text{pot}(x, n/2)$; ¿Qué esquema resulta ser más eficiente en cuanto al coste temporal?

- (a) En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.
- (b) Programación dinámica.
- (c) Divide y vencerás.

6. Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones recursiva expresa mejor su complejidad temporal para el caso general, siendo n el número de discos?
- (a) $T(n) = T(n - 1) + n$
(b) $T(n) = 2T(n - 1) + 1$
(c) $T(n) = 2T(n - 1) + n$
7. El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continue ordenado es...
- (a) ... $O(\log n)$.
(b) ... $\Omega(n^2)$.
(c) ... $O(n)$.
8. 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 recursión apropiada para el caso general?
- (a) Ninguna de las otras dos recursiones se corresponde con un esquema de divide y vencerás.
(b) $nc(n, m) = nc(n - 1, m) + nc(n, m - 1) + nc(n - 1, m - 1)$
(c) $nc(n, m) = nc(n - 1, m) * nc(n, m - 1) * nc(n - 1, m - 1)$
9. Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿qué propiedad cumple una cota optimista?
- (a)** Las otras dos opciones son ambas falsas.
(b) Asegura un ahorro en la comprobación de todas las soluciones factibles.
(c) Siempre es mayor o igual que la mejor solución posible alcanzada.
10. En el esquema de ramificación y poda, ¿qué estructura es la más adecuada si queremos realizar una exploración por niveles?
- (a)** Cola
(b) Cola de prioridad
(c) Pila

11. Dado el problema del laberinto con tres movimientos, ¿se puede aplicar un esquema de programación dinámica para obtener un camino de salida?
 - (a) No, para garantizar que se encuentra un camino de salida hay que aplicar métodos de búsqueda exhaustiva como vuelta atrás o ramificación y poda.
 - (b) 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.
 - (c) Sí, en caso de existir con este esquema siempre se puede encontrar un camino de salida
12. ¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?
 - (a) Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.
 - (b) Que el algoritmo sería más lento pues se explorarían más nodos de los necesarios.
 - (c) Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.
13. Se desea ordenar una lista enlazada de n elementos haciendo uso del algoritmo *Mergesort*. En este caso, al tratarse de una lista, la complejidad temporal asintótica de realizar la división en subproblemas resulta ser lineal con el tamaño de esa lista. ¿Cuál sería entonces el coste temporal de realizar dicha ordenación?
 - (a) Ninguna de las otras dos opciones es cierta.
 - (b) $\Theta(n \log n)$
 - (c) $\Theta(n^2)$
14. Una de las prácticas de laboratorio consistió en el cálculo empírico de la complejidad temporal promedio del algoritmo de ordenación de vectores Quicksort tomando como centinela el elemento del vector que ocupa la posición central. ¿Cuál es el orden de complejidad que se obtuvo?
 - (a) $n \log n$
 - (b) $n \log^2 n$
 - (c) n^2

15. Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿cuál sería la forma más adecuada de representar las posibles soluciones?
- (a) un par de enteros que indiquen los cortes realizados y el valor acumulado.
 - (b) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.
 - (c) un vector de booleanos.
16. Si $f \in \Omega(g_1)$ y $f \in \Omega(g_2)$ entonces
- (a) $f \in \Omega(g_1 \cdot g_2)$
 - (b) $f \in \Omega(g_1 + g_2)$
 - (c) $f \notin \Omega(\min(g_1, g_2))$
17. El esquema de vuelta atrás ...
- (a) Garantiza que encuentra la solución óptima a cualquier problema de selección discreta.
 - (b) Se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado.
 - (c) Las otras dos opciones son ambas verdaderas.
18. ¿Qué estrategia de búsqueda es a priori más apropiada en un esquema de vuelta atrás?
- (a) Explorar primero los nodos con mejor cota optimista.
 - (b) Explorar primero los nodos que están más completados.
 - (c) En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.
19. ¿Qué complejidad se obtiene a partir de la relación de recurrencia $T(n) = 8T(n/2) + n^3$ con $T(1) = O(1)$?
- (a) $O(n \log n)$
 - (b) $O(n^3 \log n)$
 - (c) $O(n^3)$

20. 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) $O(\log n)$
- (b) $O(n^2)$
- (c) $O(n)$

21. ¿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) El coste temporal asintótico en el caso medio.
- (b) Nada de interés.
- (c) El coste temporal promedio.

22. Si el coste temporal de un algoritmo es $T(n)$, ¿cuál de las siguientes situaciones es imposible?

- (a) $T(n) \in \Omega(n)$ y $T(n) \in \Theta(n^2)$
- (b) $T(n) \in \Theta(n)$ y $T(n) \in \Omega(n^2)$
- (c) $T(n) \in O(n)$ y $T(n) \in \Theta(n)$

23. ¿En ramificación y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?

- (a) Sí, aunque no es una garantía de que sea una buena estrategia de búsqueda.
- (b) Sí, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista.
- (c) No, la cota optimista sólo se utiliza para determinar si una *n-tupla* es prometedora.

24. Un algoritmo recursivo basado en el esquema *divide y vencerás...*

- (a) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.
- (b) Las otras dos opciones son ambas verdaderas.
- (c) ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a .

25. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1, 1)$ hasta la casilla (n, m) y para ello se aplica el esquema programación dinámica para obtener un algoritmo lo más eficiente posible en cuanto a complejidad temporal y espacial. ¿Cuáles serían ambas complejidades?
- (a) Temporal $\Theta(\max\{n, m\})$ y espacial $\Theta(\max\{n, m\})$
(b) Temporal $\Theta(n \times m)$ y espacial $\Theta(n \times m)$
 (c) Temporal $\Theta(n \times m)$ y espacial $\Theta(\min\{n, m\})$
26. 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 en realidad no es un algoritmo voraz pues las decisiones que se toman no deberían reconsiderarse.
(b) Que es un algoritmo voraz pero sin garantía de solucionar el problema.
 (c) Que en realidad no es un algoritmo voraz pues el criterio de selección no lo es.
27. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...
- (a) ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.
(b) ... debe recorrer siempre todo el árbol.
(c) ... no se puede usar para resolver problemas de optimización.
28. De las siguientes afirmaciones marca la que es verdadera.
- (a) En un esquema de vuelta atrás, las cotas pesimistas no tienen sentido si lo que se pretende es obtener todas las soluciones factibles.
(b) El esquema de vuelta atrás no es compatible con el uso conjunto de cotas pesimistas y optimistas.
(c) Las cotas pesimistas no son compatibles con un esquema de vuelta atrás.
29. El esquema voraz ...
- (a) Puede que no encuentre una solución pero si lo hace se garantiza que es óptima.
 (b) Las otras dos opciones son ambas falsas.
(c) Garantiza encontrar una solución a cualquier problema, aunque puede que no sea óptima.

30. 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) Programación dinámica.
- (b) Ramificación y poda.
- (c) Divide y vencerás.

31. Dada la siguiente función (donde $\max(a, b) \in \Theta(1)$):

```
float exa(vector<float>&v, vector<int>&p, int P, int i)
{
    float a, b;
    if (i>=0){
        if (p[i] <= P)
            a= v[i]+exa(v,p,P-p[i],i-1);
        else a= 0;
        b= exa(v,p,P,i-1);
        return max(a,b);
    }
    return 0;
}
```

Marcad la opción correcta.

- (a) La complejidad temporal en el mejor de los casos es $\Omega(n^2)$
 - (b) La complejidad temporal en el peor de los casos es $O(n^2)$
 - (c) La complejidad temporal en el peor de los casos es $O(2^n)$
32. 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) De una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización.
- (b) 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.
- (c) De una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización.

33. 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 todo el espacio de soluciones posibles.
- (b) Las otras dos opciones son ambas verdaderas.
- (c) ... pueden eliminar vectores que representan posibles soluciones factibles.

34. Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:

- (a) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- (b) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- (c) El valor de la mochila continua correspondiente .

35. 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) $O(2^{\log n}) \subset O(n^2)$
- (b) $\Theta(n) \subset \Theta(n^2)$
- (c) $n + n \log n \in \Omega(n)$

36. 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 en el mejor de los casos es $\Omega(n)$
- (b) La complejidad temporal exacta es $\Theta(n \log n)$
- (c) La complejidad temporal en el mejor de los casos es $\Omega(1)$

37. 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 en adelante todas las casillas del laberinto son accesibles.
- (b) Suponer que ya no se van a realizar más movimientos.
- (c) Las otras dos estrategias son ambas válidas.

38. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.

- (a) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
- (b) Lo más importante es conseguir una cota pesimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.
- (c) 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.

39. 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 en el mejor de los casos es $\Omega(1)$
 - (b) La complejidad temporal exacta es $\Theta(n^2)$
 - (c) La complejidad temporal en el mejor de los casos es $\Omega(n)$
40. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?
- (a) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
 - (b) Vuelta atrás, es el esquema más eficiente para este problema.
 - (c) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.

Respuestas para la modalidad 4

1. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1, 1)$ hasta la casilla (n, m) y para ello se aplica un esquema de divide y vencerás. ¿Cuál sería la recurrencia apropiada para el caso general?

- (a) $nc(n, m) = nc(n - 1, m) + nc(n, m - 1) + nc(n - 1, m - 1)$
- (b) $nc(n, m) = nc(n - 1, m) * nc(n, m - 1) * nc(n - 1, m - 1)$
- (c) Ninguna de las otras dos recurrencias se corresponde con un esquema de divide y vencerás.

2. 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) $O(\log n)$

3. Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es par y que se utilizará la siguiente recurrencia: $\text{pot}(x, n) = \text{pot}(x, n/2) * \text{pot}(x, n/2)$; ¿Qué esquema resulta ser más eficiente en cuanto al coste temporal?

- (a) En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.
- (b) Programación dinámica.
- (c) Divide y vencerás.

4. ¿Qué estrategia de búsqueda es a priori más apropiada en un esquema de vuelta atrás?

- (a) Explorar primero los nodos que están más completados.
- (b) En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.
- (c) Explorar primero los nodos con mejor cota optimista.

5. Se desea ordenar una lista enlazada de n elementos haciendo uso del algoritmo *Mergesort*. En este caso, al tratarse de una lista, la complejidad temporal asintótica de realizar la división en subproblemas resulta ser lineal con el tamaño de esa lista. ¿Cuál sería entonces el coste temporal de realizar dicha ordenación?
- (a) $\Theta(n \log n)$
 - (b) $\Theta(n^2)$
 - (c) Ninguna de las otras dos opciones es cierta.
6. 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.
7. ¿Qué se deduce de $f(n)$ y $g(n)$ si se cumple $\lim_{n \rightarrow \infty} (f(n)/g(n)) = k$, con $k \neq 0$?
- (a) $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
 - (b) $f(n) \in O(g(n))$ pero $g(n) \notin O(f(n))$
 - (c) $g(n) \in O(f(n))$ pero $f(n) \notin O(g(n))$
8. 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.
9. Si $f \in \Omega(g_1)$ y $f \in \Omega(g_2)$ entonces
- (a) $f \in \Omega(g_1 + g_2)$
 - (b) $f \notin \Omega(\min(g_1, g_2))$
 - (c) $f \in \Omega(g_1 \cdot g_2)$
10. El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continue ordenado es...
- (a) ... $\Omega(n^2)$.
 - (b) ... $O(n)$.
 - (c) ... $O(\log n)$.

11. ¿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.
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) $\Theta(n) \subset \Theta(n^2)$
 - (b) $n + n \log n \in \Omega(n)$
 - (c) $O(2^{\log n}) \subset O(n^2)$
13. 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.
14. ¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?
 - (a) Que el algoritmo sería más lento pues se explorarían más nodos de los necesarios.
 - (b) Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.
 - (c) Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.

15. Dada la siguiente función:

```
int exa (vector <int>& v){
    int j, i=1, n=v.size();

    if (n>1) do{
        int x = v[i];
        for (j=i; j >0 and v[j-1] >x; j--)
            v[j]=v[j-1];
        v[j]=x;
        i++;
    } while (i<n);
    return 0;
}
```

Marcad la opción correcta.

- (a) La complejidad temporal exacta es $\Theta(n^2)$
- (b) La complejidad temporal en el mejor de los casos es $\Omega(n)$
- (c) La complejidad temporal en el mejor de los casos es $\Omega(1)$

16. ¿Qué complejidad se obtiene a partir de la relación de recurrencia $T(n) = 8T(n/2) + n^3$ con $T(1) = O(1)$?

- (a) $O(n^3 \log n)$
- (b) $O(n^3)$
- (c) $O(n \log n)$

17. ¿En ramificación y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?

- (a) Sí, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista.
- (b) No, la cota optimista sólo se utiliza para determinar si una n -tupla es prometedora.
- (c) Sí, aunque no es una garantía de que sea una buena estrategia de búsqueda.

18. 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.

19. 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.
20. 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.
21. Una de las prácticas de laboratorio consistió en el cálculo empírico de la complejidad temporal promedio del algoritmo de ordenación de vectores Quicksort tomando como centinela el elemento del vector que ocupa la posición central. ¿Cuál es el orden de complejidad que se obtuvo?
- (a) $n \log^2 n$
- (b) n^2
- (c) $n \log n$
22. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1, 1)$ hasta la casilla (n, m) y para ello se aplica el esquema programación dinámica para obtener un algoritmo lo más eficiente posible en cuanto a complejidad temporal y espacial. ¿Cuáles serían ambas complejidades?
- (a) Temporal $\Theta(n \times m)$ y espacial $\Theta(n \times m)$
- (b) Temporal $\Theta(n \times m)$ y espacial $\Theta(\min\{n, m\})$
- (c) Temporal $\Theta(\max\{n, m\})$ y espacial $\Theta(\max\{n, m\})$
23. 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.

24. Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿qué propiedad cumple una cota optimista?
- (a) Asegura un ahorro en la comprobación de todas las soluciones factibles.
 - (b) Siempre es mayor o igual que la mejor solución posible alcanzada.
 - (c) Las otras dos opciones son ambas falsas.
25. Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones recursiva 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) $T(n) = T(n - 1) + n$
26. 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.
27. 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.
28. 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.

29. Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?
- una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.
 - un vector de booleanos.
 - un par de enteros que indiquen los cortes realizados y el valor acumulado.
30. ¿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);
}
```
- $\Theta(n)$
  - $\Theta(n^2)$
  - $\Theta(n \times m)$
31. 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?
- Que es un algoritmo voraz pero sin garantía de solucionar el problema.
  - Que en realidad no es un algoritmo voraz pues el criterio de selección no lo es.
  - Que en realidad no es un algoritmo voraz pues las decisiones que se toman no deberían reconsiderarse.
32. En el esquema de ramificación y poda, ¿qué estructura es la más adecuada si queremos realizar una exploración por niveles?
- Cola de prioridad
  - Pila
  - Cola

33. 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.
34. Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?
- (a) Que se podría explorar menos nodos de los necesarios.
  - (b) Que se podría podar el nodo que conduce a la solución óptima.
  - (c) Que se podría explorar más nodos de los necesarios.
35. Si el coste temporal de un algoritmo es  $T(n)$ , ¿cuál de las siguientes situaciones es imposible?
- (a)  $T(n) \in \Theta(n)$  y  $T(n) \in \Omega(n^2)$
  - (b)  $T(n) \in O(n)$  y  $T(n) \in \Theta(n)$
  - (c)  $T(n) \in \Omega(n)$  y  $T(n) \in \Theta(n^2)$
36. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
- (a) Lo más importante es conseguir una cota pesimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.
  - (b) Lo más adecuado sería usar una técnica de ramificación y poda ya que es muy importante el orden en el que se exploran las soluciones parciales.
  - (c) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
37. 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.

38. Dada la siguiente función (donde  $\max(a,b) \in \Theta(1)$ ):

```
float exa(vector<float>&v, vector<int>&p, int P, int i)
{
 float a, b;
 if (i>=0){
 if (p[i] <= P)
 a= v[i]+exa(v,p,P-p[i],i-1);
 else a= 0;
 b= exa(v,p,P,i-1);
 return max(a,b);
 }
 return 0;
}
```

Marcad la opción correcta.

- (a) La complejidad temporal en el peor de los casos es  $O(n^2)$
- (b) La complejidad temporal en el peor de los casos es  $O(2^n)$
- (c) La complejidad temporal en el mejor de los casos es  $\Omega(n^2)$

39. Dada la siguiente función:

```
int exa (vector <int>& v) {
 int i,sum=0, n=v.size();

 if (n>0){
 int j=n;
 while (sum<100){
 j=j/2;
 sum=0;
 for (i=j;i<n;i++)
 sum+=v[i];
 if (j==0) sum=100;
 }
 return j;
 }
 else return -1;
}
```

Marcad la opción correcta.

- (a) La complejidad temporal exacta es  $\Theta(n \log n)$
- (b) La complejidad temporal en el mejor de los casos es  $\Omega(1)$
- (c) La complejidad temporal en el mejor de los casos es  $\Omega(n)$

40. Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in O(n)$ , ¿en cuál de estos tres casos nos podemos encontrar?

- (a) Las otras dos opciones son ambas ciertas.
- (b)  $g(n) = \log n$
- (c)  $g(n) = \sqrt{n}$

## Respuestas para la modalidad 5

1. Dado el problema del laberinto con tres movimientos, ¿se puede aplicar un esquema de programación dinámica para obtener un camino de salida?
  - (a) No, con este esquema se puede conocer el número total de caminos distintos que conducen a la salida pero no se puede saber la composición de ninguno de ellos.
  - (b) Sí, en caso de existir con este esquema siempre se puede encontrar un camino de salida
  - (c) No, para garantizar que se encuentra un camino de salida hay que aplicar métodos de búsqueda exhaustiva como vuelta atrás o ramificación y poda.
2. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...
  - (a) ... debe recorrer siempre todo el árbol.
  - (b) ... no se puede usar para resolver problemas de optimización.
  - (c) ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.
3. Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:
  - (a) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
  - (b) El valor de la mochila continua correspondiente .
  - (c) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
4. El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continúe ordenado es...
  - (a) ...  $\Omega(n^2)$ .
  - (b) ...  $O(n)$ .
  - (c) ...  $O(\log n)$ .
5. ¿En ramificación y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?
  - (a) Sí, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista.
  - (b) No, la cota optimista sólo se utiliza para determinar si una  $n$ -tupla es prometedora.
  - (c) Sí, aunque no es una garantía de que sea una buena estrategia de búsqueda.

6. ¿Qué estrategia de búsqueda es a priori más apropiada en un esquema de vuelta atrás?

- (a) Explorar primero los nodos que están más completados.
- (b) En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.
- (c) Explorar primero los nodos con mejor cota optimista.

7. Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in O(n)$ , ¿en cuál de estos tres casos nos podemos encontrar?

- (a) Las otras dos opciones son ambas ciertas.
- (b)  $g(n) = \log n$
- (c)  $g(n) = \sqrt{n}$

8. Si  $f \in \Omega(g_1)$  y  $f \in \Omega(g_2)$  entonces

- (a)  $f \in \Omega(g_1 + g_2)$
- (b)  $f \notin \Omega(\min(g_1, g_2))$
- (c)  $f \in \Omega(g_1 \cdot g_2)$

9. Dada la siguiente función:

```
int exa (string & cad, int pri, int ult){

 if (pri>=ult)
 return 1;
 else
 if (cad[pri]==cad[ult])
 return exa(cad, pri+1, ult-1);
 else
 return 0;
}
```

¿Cuál es su complejidad temporal asintótica?

- (a)  $O(n^2)$
- (b)  $O(n)$
- (c)  $O(\log n)$

10. El esquema de vuelta atrás ...

- (a) Se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado.
- (b) Las otras dos opciones son ambas verdaderas.
- (c) Garantiza que encuentra la solución óptima a cualquier problema de selección discreta.

11. En el esquema de ramificación y poda, ¿qué estructura es la más adecuada si queremos realizar una exploración por niveles?
  - (a) Cola de prioridad
  - (b) Pila
  - (c) Cola
12. ¿Qué nos proporciona la media entre el coste temporal asintótico (o complejidad temporal) en el peor caso y el coste temporal asintótico en el mejor caso?
  - (a) Nada de interés.
  - (b) El coste temporal promedio.
  - (c) El coste temporal asintótico en el caso medio.
13. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial  $(1, 1)$  hasta la casilla  $(n, m)$  y para ello se aplica un esquema de programación dinámica. En cuanto a la complejidad temporal, ¿cuál es la mejora de la versión recursiva con memoización frente a la recursiva ingenua que se obtiene a partir del esquema divide y vencerás?
  - (a) La mejora no está garantizada puesto que la versión recursiva con memoización podría ser peor que la obtenida a partir del esquema divide y vencerás.
  - (b) De una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización.
  - (c) De una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización.
14. ¿Qué complejidad se obtiene a partir de la relación de recurrencia  $T(n) = 8T(n/2) + n^3$  con  $T(1) = O(1)$ ?
  - (a)  $O(n^3 \log n)$
  - (b)  $O(n^3)$
  - (c)  $O(n \log n)$

15. Dada la siguiente función:

```
int exa (vector <int>& v){
 int j, i=1, n=v.size();

 if (n>1) do{
 int x = v[i];
 for (j=i; j >0 and v[j-1] >x; j--)
 v[j]=v[j-1];
 v[j]=x;
 i++;
 } while (i<n);
 return 0;
}
```

Marcad la opción correcta.

- (a) La complejidad temporal exacta es  $\Theta(n^2)$
- (b) La complejidad temporal en el mejor de los casos es  $\Omega(n)$
- (c) La complejidad temporal en el mejor de los casos es  $\Omega(1)$

16. El esquema voraz ...

- (a) Las otras dos opciones son ambas falsas.
- (b) Garantiza encontrar una solución a cualquier problema, aunque puede que no sea óptima.
- (c) Puede que no encuentre una solución pero si lo hace se garantiza que es óptima.

17. Cuando la descomposición de un problema da lugar a subproblemas de tamaño similar al original, muchos de los cuales se repiten, ¿qué esquema es a priori más apropiado?

- (a) Ramificación y poda.
- (b) Divide y vencerás.
- (c) Programación dinámica.

18. Dada la siguiente función (donde  $\max(a,b) \in \Theta(1)$ ):

```
float exa(vector<float>&v, vector<int>&p, int P, int i)
{
 float a, b;
 if (i>=0){
 if (p[i] <= P)
 a= v[i]+exa(v,p,P-p[i],i-1);
 else a= 0;
 b= exa(v,p,P,i-1);
 return max(a,b);
 }
 return 0;
}
```

Marcad la opción correcta.

- (a) La complejidad temporal en el peor de los casos es  $O(n^2)$
- (b) La complejidad temporal en el peor de los casos es  $O(2^n)$
- (c) La complejidad temporal en el mejor de los casos es  $\Omega(n^2)$

19. ¿Cuál sería la complejidad temporal de la siguiente función tras aplicar programación dinámica?

```
double f(int n, int m) {
 if(n == 0) return 1;
 return m * f(n-1,m) * f(n-2,m);
}
```

- (a)  $\Theta(n)$
- (b)  $\Theta(n^2)$
- (c)  $\Theta(n \times m)$

20. Dado el problema del laberinto con tres movimientos, se pretende conocer la longitud del camino de salida más corto. Para ello se aplica el esquema voraz con un criterio de selección que consiste en elegir primero el movimiento *Este* siempre que la casilla sea accesible. Si no lo es se descarta ese movimiento y se prueba con *Sureste* y por último, si este tampoco es posible, se escoge el movimiento *Sur*. ¿Qué se puede decir del algoritmo obtenido?

- (a) Que es un algoritmo voraz pero sin garantía de solucionar el problema.
- (b) Que en realidad no es un algoritmo voraz pues el criterio de selección no lo es.
- (c) Que en realidad no es un algoritmo voraz pues las decisiones que se toman no deberían reconsiderarse.

21. Si el coste temporal de un algoritmo es  $T(n)$ , ¿cuál de las siguientes situaciones es imposible?
- (a)  $T(n) \in \Theta(n)$  y  $T(n) \in \Omega(n^2)$
  - (b)  $T(n) \in O(n)$  y  $T(n) \in \Theta(n)$
  - (c)  $T(n) \in \Omega(n)$  y  $T(n) \in \Theta(n^2)$
22. Se desea resolver el problema de la potencia enésima ( $x^n$ ), asumiendo que  $n$  es par y que se utilizará la siguiente recurrencia:  $\text{pot}(x, n) = \text{pot}(x, n/2) * \text{pot}(x, n/2)$ ; ¿Qué esquema resulta ser más eficiente en cuanto al coste temporal?
- (a) En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.
  - (b) Programación dinámica.
  - (c) Divide y vencerás.
23. De las siguientes afirmaciones marca la que es verdadera.
- (a) El esquema de vuelta atrás no es compatible con el uso conjunto de cotas pesimistas y optimistas.
  - (b) Las cotas pesimistas no son compatibles con un esquema de vuelta atrás.
  - (c) En un esquema de vuelta atrás, las cotas pesimistas no tienen sentido si lo que se pretende es obtener todas las soluciones factibles.
24. Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿qué propiedad cumple una cota optimista?
- (a) Asegura un ahorro en la comprobación de todas las soluciones factibles.
  - (b) Siempre es mayor o igual que la mejor solución posible alcanzada.
  - (c) Las otras dos opciones son ambas falsas.
25. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial  $(1, 1)$  hasta la casilla  $(n, m)$  y para ello se aplica el esquema programación dinámica para obtener un algoritmo lo más eficiente posible en cuanto a complejidad temporal y espacial. ¿Cuáles serían ambas complejidades?
- (a) Temporal  $\Theta(n \times m)$  y espacial  $\Theta(n \times m)$
  - (b) Temporal  $\Theta(n \times m)$  y espacial  $\Theta(\min\{n, m\})$
  - (c) Temporal  $\Theta(\max\{n, m\})$  y espacial  $\Theta(\max\{n, m\})$
26. ¿Qué se deduce de  $f(n)$  y  $g(n)$  si se cumple  $\lim_{n \rightarrow \infty} (f(n)/g(n)) = k$ , con  $k \neq 0$ ?
- (a)  $f(n) \in O(g(n))$  y  $g(n) \in O(f(n))$
  - (b)  $f(n) \in O(g(n))$  pero  $g(n) \notin O(f(n))$
  - (c)  $g(n) \in O(f(n))$  pero  $f(n) \notin O(g(n))$

27. Se desea obtener todas las permutaciones de una lista compuesta por  $n$  elementos. ¿Qué esquema es el más adecuado?
- (a) Vuelta atrás, es el esquema más eficiente para este problema.
  - (b) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
  - (c) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
28. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial  $(1, 1)$  hasta la casilla  $(n, m)$  y para ello se aplica un esquema de divide y vencerás. ¿Cuál sería la recurrencia apropiada para el caso general?
- (a)  $nc(n, m) = nc(n - 1, m) + nc(n, m - 1) + nc(n - 1, m - 1)$
  - (b)  $nc(n, m) = nc(n - 1, m) * nc(n, m - 1) * nc(n - 1, m - 1)$
  - (c) Ninguna de las otras dos recurrencias se corresponde con un esquema de divide y vencerás.
29. Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?
- (a) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.
  - (b) un vector de booleanos.
  - (c) un par de enteros que indiquen los cortes realizados y el valor acumulado.
30. De las siguientes expresiones, o bien dos son verdaderas y una es falsa, o bien dos son falsas y una es verdadera. Marca la que (en este sentido) es distinta a las otras dos.
- (a)  $\Theta(n) \subset \Theta(n^2)$
  - (b)  $n + n \log n \in \Omega(n)$
  - (c)  $O(2^{\log n}) \subset O(n^2)$
31. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...
- (a) Las otras dos opciones son ambas verdaderas.
  - (b) ... pueden eliminar vectores que representan posibles soluciones factibles.
  - (c) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.

32. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
- (a) Lo más importante es conseguir una cota pesimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.
- (b) Lo más adecuado sería usar una técnica de ramificación y poda ya que es muy importante el orden en el que se exploran las soluciones parciales.
- (c) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
33. Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?
- (a) Que se podría explorar menos nodos de los necesarios.
- (b) Que se podría podar el nodo que conduce a la solución óptima.
- (c) Que se podría explorar más nodos de los necesarios.
34. Un algoritmo recursivo basado en el esquema *divide y vencerás*...
- (a) Las otras dos opciones son ambas verdaderas.
- (b) ... alcanza su máxima eficiencia cuando el problema de tamaño  $n$  se divide en  $a$  problemas de tamaño  $n/a$ .
- (c) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.
35. Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones recursiva expresa mejor su complejidad temporal para el caso general, siendo  $n$  el número de discos?
- (a)  $T(n) = 2T(n - 1) + 1$
- (b)  $T(n) = 2T(n - 1) + n$
- (c)  $T(n) = T(n - 1) + n$
36. Una de las prácticas de laboratorio consistió en el cálculo empírico de la complejidad temporal promedio del algoritmo de ordenación de vectores Quicksort tomando como centinela el elemento del vector que ocupa la posición central. ¿Cuál es el orden de complejidad que se obtuvo?
- (a)  $n \log^2 n$
- (b)  $n^2$
- (c)  $n \log n$

37. Se desea ordenar una lista enlazada de  $n$  elementos haciendo uso del algoritmo *Mergesort*. En este caso, al tratarse de una lista, la complejidad temporal asintótica de realizar la división en subproblemas resulta ser lineal con el tamaño de esa lista. ¿Cuál sería entonces el coste temporal de realizar dicha ordenación?

- (a)  $\Theta(n \log n)$
- (b)  $\Theta(n^2)$
- (c) Ninguna de las otras dos opciones es cierta.

38. Dada la siguiente función:

```
int exa (vector <int>& v) {
 int i,sum=0, n=v.size();

 if (n>0){
 int j=n;
 while (sum<100){
 j=j/2;
 sum=0;
 for (i=j;i<n;i++)
 sum+=v[i];
 if (j==0) sum=100;
 }
 return j;
 }
 else return -1;
}
```

Marcad la opción correcta.

- (a) La complejidad temporal exacta es  $\Theta(n \log n)$
- (b) La complejidad temporal en el mejor de los casos es  $\Omega(1)$
- (c) La complejidad temporal en el mejor de los casos es  $\Omega(n)$

39. Dado el problema del laberinto con tres movimientos, ¿cuál de las estrategias siguientes proveería de una cota optimista para ramificación y poda?

- (a) Suponer que ya no se van a realizar más movimientos.
- (b) Las otras dos estrategias son ambas válidas.
- (c) Suponer que en adelante todas las casillas del laberinto son accesibles.

40. ¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?

- (a) Que el algoritmo sería más lento pues se explorarían más nodos de los necesarios.
- (b) Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.
- (c) Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.

## Respuestas para la modalidad 6

1. 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) De una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización.
  - (b) 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.
  - (c) De una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización.
2. ¿Qué complejidad se obtiene a partir de la relación de recurrencia  $T(n) = 8T(n/2) + n^3$  con  $T(1) = O(1)$ ?
  - (a)  $O(n \log n)$
  - (b)  $O(n^3 \log n)$
  - (c)  $O(n^3)$
3. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
  - (a) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
  - (b) Lo más importante es conseguir una cota pesimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.
  - (c) 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.
4. 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) Ninguna de las otras dos recurrencias se corresponde con un esquema de divide y vencerás.
  - (b)  $nc(n, m) = nc(n - 1, m) + nc(n, m - 1) + nc(n - 1, m - 1)$
  - (c)  $nc(n, m) = nc(n - 1, m) * nc(n, m - 1) * nc(n - 1, m - 1)$

5. ¿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) El coste temporal asintótico en el caso medio.
  - (b) Nada de interés.
  - (c) El coste temporal promedio.
6. ¿En ramificación y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?
- (a) Sí, aunque no es una garantía de que sea una buena estrategia de búsqueda.
  - (b) Sí, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista.
  - (c) No, la cota optimista sólo se utiliza para determinar si una *n-tupla* es prometedora.
7. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...
- (a) ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.
  - (b) ... debe recorrer siempre todo el árbol.
  - (c) ... no se puede usar para resolver problemas de optimización.
8. 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 en el mejor de los casos es $\Omega(1)$
 - (b) La complejidad temporal exacta es $\Theta(n^2)$
 - (c) La complejidad temporal en el mejor de los casos es $\Omega(n)$
9. El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continúe ordenado es...
- (a) ... $O(\log n)$.
 - (b) ... $\Omega(n^2)$.
 - (c) ... $O(n)$.

10. ¿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) $\Theta(n \times m)$
- (b) $\Theta(n)$
- (c) $\Theta(n^2)$

11. Un algoritmo recursivo basado en el esquema *divide y vencerás...*

- (a) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.
- (b) Las otras dos opciones son ambas verdaderas.
- (c) ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a .

12. De las siguientes afirmaciones marca la que es verdadera.

- (a) En un esquema de vuelta atrás, las cotas pesimistas no tienen sentido si lo que se pretende es obtener todas las soluciones factibles.
- (b) El esquema de vuelta atrás no es compatible con el uso conjunto de cotas pesimistas y optimistas.
- (c) Las cotas pesimistas no son compatibles con un esquema de vuelta atrás.

13. 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) $T(n) = T(n - 1) + n$
- (b) $T(n) = 2T(n - 1) + 1$
- (c) $T(n) = 2T(n - 1) + n$

14. Se desea ordenar una lista enlazada de n elementos haciendo uso del algoritmo *Mergesort*. En este caso, al tratarse de una lista, la complejidad temporal asintótica de realizar la división en subproblemas resulta ser lineal con el tamaño de esa lista. ¿Cuál sería entonces el coste temporal de realizar dicha ordenación?

- (a) Ninguna de las otras dos opciones es cierta.
- (b) $\Theta(n \log n)$
- (c) $\Theta(n^2)$

15. El esquema voraz ...

- (a) Puede que no encuentre una solución pero si lo hace se garantiza que es óptima.
- (b) Las otras dos opciones son ambas falsas.
- (c) Garantiza encontrar una solución a cualquier problema, aunque puede que no sea óptima.

16. Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:

- (a) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- (b) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- (c) El valor de la mochila continua correspondiente .

17. Dada la siguiente función (donde $\max(a, b) \in \Theta(1)$):

```
float exa(vector<float>&v, vector<int>&p, int P, int i)
{
    float a, b;
    if (i>=0){
        if (p[i] <= P)
            a= v[i]+exa(v,p,P-p[i],i-1);
        else a= 0;
        b= exa(v,p,P,i-1);
        return max(a,b);
    }
    return 0;
}
```

Marcad la opción correcta.

- (a) La complejidad temporal en el mejor de los casos es $\Omega(n^2)$
 - (b) La complejidad temporal en el peor de los casos es $O(n^2)$
 - (c) La complejidad temporal en el peor de los casos es $O(2^n)$
18. 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) Programación dinámica.
 - (b) Ramificación y poda.
 - (c) Divide y vencerás.

19. ¿Qué estrategia de búsqueda es a priori más apropiada en un esquema de vuelta atrás?
- (a) Explorar primero los nodos con mejor cota optimista.
 - (b) Explorar primero los nodos que están más completados.
 - (c) En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.
20. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1, 1)$ hasta la casilla (n, m) y para ello se aplica el esquema programación dinámica para obtener un algoritmo lo más eficiente posible en cuanto a complejidad temporal y espacial. ¿Cuáles serían ambas complejidades?
- (a) Temporal $\Theta(\max\{n, m\})$ y espacial $\Theta(\max\{n, m\})$
 - (b) Temporal $\Theta(n \times m)$ y espacial $\Theta(n \times m)$
 - (c) Temporal $\Theta(n \times m)$ y espacial $\Theta(\min\{n, m\})$
21. Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?
- (a) Que se podría explorar más nodos de los necesarios.
 - (b) Que se podría explorar menos nodos de los necesarios.
 - (c) Que se podría podar el nodo que conduce a la solución óptima.
22. 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) $O(2^{\log n}) \subset O(n^2)$
 - (b) $\Theta(n) \subset \Theta(n^2)$
 - (c) $n + n \log n \in \Omega(n)$
23. Una de las prácticas de laboratorio consistió en el cálculo empírico de la complejidad temporal promedio del algoritmo de ordenación de vectores Quicksort tomando como centinela el elemento del vector que ocupa la posición central. ¿Cuál es el orden de complejidad que se obtuvo?
- (a) $n \log n$
 - (b) $n \log^2 n$
 - (c) n^2

24. Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?
- (a) un par de enteros que indiquen los cortes realizados y el valor acumulado.
 - (b) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.
 - (c) un vector de booleanos.
25. Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es par y que se utilizará la siguiente recurrencia: $\text{pot}(x, n) = \text{pot}(x, n/2) * \text{pot}(x, n/2)$; ¿Qué esquema resulta ser más eficiente en cuanto al coste temporal?
- (a) En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.
 - (b) Programación dinámica.
 - (c) Divide y vencerás.
26. Dado el problema del laberinto con tres movimientos, ¿se puede aplicar un esquema de programación dinámica para obtener un camino de salida?
- (a) No, para garantizar que se encuentra un camino de salida hay que aplicar métodos de búsqueda exhaustiva como vuelta atrás o ramificación y poda.
 - (b) 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.
 - (c) Sí, en caso de existir con este esquema siempre se puede encontrar un camino de salida.
27. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?
- (a) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
 - (b) Vuelta atrás, es el esquema más eficiente para este problema.
 - (c) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.

28. El esquema de vuelta atrás ...

- (a) Garantiza que encuentra la solución óptima a cualquier problema de selección discreta.
- (b) Se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado.
- (c) Las otras dos opciones son ambas verdaderas.

29. 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 en adelante todas las casillas del laberinto son accesibles.
- (b) Suponer que ya no se van a realizar más movimientos.
- (c) Las otras dos estrategias son ambas válidas.

30. 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) $O(\log n)$
- (b) $O(n^2)$
- (c) $O(n)$

31. 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 todo el espacio de soluciones posibles.
- (b) Las otras dos opciones son ambas verdaderas.
- (c) ... pueden eliminar vectores que representan posibles soluciones factibles.

32. 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 en realidad no es un algoritmo voraz pues las decisiones que se toman no deberían reconsiderarse.
- (b) Que es un algoritmo voraz pero sin garantía de solucionar el problema.
- (c) Que en realidad no es un algoritmo voraz pues el criterio de selección no lo es.

33. Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si $T(n) \in O(n)$, ¿en cuál de estos tres casos nos podemos encontrar?

- (a) $g(n) = \sqrt{n}$
- (b) Las otras dos opciones son ambas ciertas.
- (c) $g(n) = \log n$

34. Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿qué propiedad cumple una cota optimista?

- (a) Las otras dos opciones son ambas falsas.
- (b) Asegura un ahorro en la comprobación de todas las soluciones factibles.
- (c) Siempre es mayor o igual que la mejor solución posible alcanzada.

35. Si $f \in \Omega(g_1)$ y $f \in \Omega(g_2)$ entonces

- (a) $f \in \Omega(g_1 \cdot g_2)$
- (b) $f \in \Omega(g_1 + g_2)$
- (c) $f \notin \Omega(\min(g_1, g_2))$

36. 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 en el mejor de los casos es $\Omega(n)$
- (b) La complejidad temporal exacta es $\Theta(n \log n)$
- (c) La complejidad temporal en el mejor de los casos es $\Omega(1)$

37. ¿Qué se deduce de $f(n)$ y $g(n)$ si se cumple $\lim_{n \rightarrow \infty} (f(n)/g(n)) = k$, con $k \neq 0$?

- (a) $g(n) \in O(f(n))$ pero $f(n) \notin O(g(n))$
- (b) $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
- (c) $f(n) \in O(g(n))$ pero $g(n) \notin O(f(n))$

38. Si el coste temporal de un algoritmo es $T(n)$, ¿cuál de las siguientes situaciones es imposible?

- (a) $T(n) \in \Omega(n)$ y $T(n) \in \Theta(n^2)$
- (b) $T(n) \in \Theta(n)$ y $T(n) \in \Omega(n^2)$
- (c) $T(n) \in O(n)$ y $T(n) \in \Theta(n)$

39. ¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?

- (a) Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.
- (b) Que el algoritmo sería más lento pues se explorarían más nodos de los necesarios.
- (c) Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.

40. En el esquema de ramificación y poda, ¿qué estructura es la más adecuada si queremos realizar una exploración por niveles?

- (a) Cola
- (b) Cola de prioridad
- (c) Pila

Preguntas:

1. 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 recorrida moviéndose al azar por el grafo hasta llegar (por azar) a la ciudad destino.
(b) Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.
(c) Suponer que la siguiente ciudad más cercana es ya la ciudad de destino.
2. 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) No se trataría de ninguna cota puesto que es posible que esa heurística no encuentre una solución factible.
→(b) Una cota optimista.
(c) Una cota pesimista.
3. Dado un vector de tamaño n de números enteros. ¿Con qué complejidad temporal se puede determinar si sus elementos están dispuestos formando un montículo de mínimos?
→(a) $O(n)$
(b) $O(\log n)$
(c) $O(1)$
4. Un problema de decisión en el que no hay función objetivo ... (marca la opción correcta)
(a) ... solo puede ser resuelto de manera eficiente mediante un algoritmo voraz.
(b) ... no puede ser resuelto mediante la técnica de ramificación y poda.
(c) ... puede que tenga solución eficiente mediante programación dinámica.

5. Haciendo uso de la función *Merge* del algoritmo *Mergesort* se quiere mezclar k vectores ordenados de n elementos cada uno y obtener un único vector de kn elementos. Para ello primero se mezclan los dos primeros vectores, luego el resultado se mezcla con el tercero y a su vez, este resultado se mezcla con el cuarto y así hasta llegar al k -ésimo. ¿Cuál será la complejidad del algoritmo?

- (a) $\Theta(n \cdot k^2)$
- (b) $\Theta(n \cdot k)$
- (c) $\Theta(n^2 \cdot k)$

6. Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f( int x, int y ) {
    if( x <= y )
        return 1;
    return x + f(x-1,y);
}
```

¿Cuál son las mejores complejidades temporal y espacial que se pueden conseguir?

- (a) Temporal $O(x^2)$ y espacial $O(x)$
- (b) $O(x)$, tanto temporal como espacial.
- (c) Temporal $O(x)$ y espacial $O(1)$

7. Una de estas tres afirmaciones es falsa. ¿Cuál?

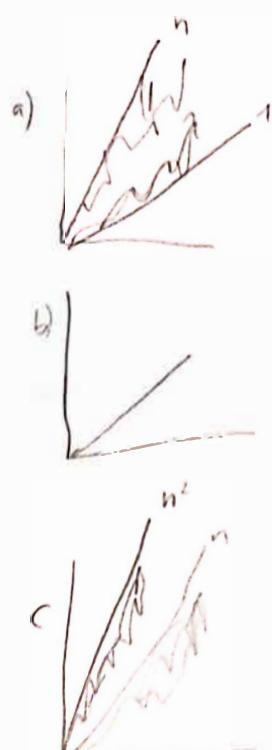
- (a) Un algoritmo voraz puede no encontrar la solución óptima de un problema de selección discreta.
- (b) El esquema de ramificación y poda no garantiza que la complejidad temporal de resolución de un problema de selección discreta no sea exponencial.
- (c) Los algoritmos voraces no sirven para resolver problemas de selección discreta.

8. Una de estas tres situaciones no es posible. ¿Cuál es?

- (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 $f(n) \in O(n^2)$

9. De las siguientes tres afirmaciones, una es cierta y dos falsas, o bien una es falsa y dos son ciertas. Marca la que en ese sentido es diferente a las otras dos.

- (a) $O(2^{\log_2(n)}) \subseteq O(n^2) \subset O(2^n)$ $n \leq n^2 < 2^n$
- (b) $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$ $n^2 < n \ll 2^n$
- (c) $O(4^{\log_2(n)}) \subseteq O(n^2) \subset O(2^n)$ $2n \leq n^2 < 2^n$



10. ¿Cuál de las siguientes expresiones es falsa?
- $\Theta(\log_2(n)) = \Theta(\log_3(n))$
 - $\Theta(\log^2(n)) = \Theta(\log^3(n))$
 - $\Theta(\log(n^2)) = \Theta(\log(n^3))$
11. Una empresa de mensajería tiene n repartidores con distintos tiempos de entrega según el tipo de envío. Se trata de asignar los próximos n envíos, uno a cada repartidor, minimizando el tiempo total de todos los envíos. Para ello se conoce de antemano una tabla de tiempos en la que el valor t_{ij} corresponde al tiempo que emplea el repartidor i en realizar el envío j . De entre las estrategias que se citan, ¿cuál sería la eficiente para resolver el problema?
- Algoritmo voraz.
 - Vuelta atrás.
 - Ramificación y poda.
12. Para que la complejidad de un algoritmo presente caso mejor y peor distintos ...
- ... es condición suficiente que existan instancias distintas del problema con el mismo tamaño.
 - ... es condición necesaria y suficiente que existan instancias distintas del problema con el mismo tamaño.
 - ... es condición necesaria que existan instancias distintas del problema con el mismo tamaño.
13. Si $\lim_{n \rightarrow \infty} (f(n)/n^2) = 0$, ¿Cuál de estas tres afirmaciones es la única que podría ser cierta?
- $f(n) \in \Theta(n^2)$
 - $f(n) \in \Theta(n^3)$
 - $f(n) \in \Theta(n)$
14. De las siguientes tres afirmaciones, una es cierta y dos falsas, o bien una es falsa y dos son ciertas. Marca la que en ese sentido es diferente a las otras dos.
- Para que un problema tenga solución mediante programación dinámica es condición necesaria que pueda resolverse mediante divide y vencerás.
 - Todo problema que tiene solución mediante divide y vencerás también la tendrá mediante programación dinámica.
 - Todo problema que tiene solución mediante ramificación y poda también la tendrá mediante programación dinámica.

15. ¿En qué caso la complejidad temporal de *quicksort* es la misma que la del algoritmo de ordenación por inserción?

- (a) En el caso mejor.
- (b) En el caso peor.
- (c) En ningún caso.

16. ¿Cuál es la definición correcta de $O(f)$?

- (a) $O(f) = \{g : \mathbb{N} \rightarrow \mathbb{R}^+ \mid \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \leq cf(n)\}$
- (b) $O(f) = \{g : \mathbb{N} \rightarrow \mathbb{R}^+ \mid \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \leq cg(n)\}$
- (c) $O(f) = \{g : \mathbb{N} \rightarrow \mathbb{R}^+ \mid \forall c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \leq cf(n)\}$

17. ¿Cuál es la complejidad temporal de la siguiente función?

```
int f(int n){
    int k=0;
    for (int i = 1; i <n; i*=2)
        for (int j=i; j >0; j-=2)
            k++;
    return k;
}
```

- (a) $\Theta(n^2)$
- (b) $\Theta(n \log n)$
- (c) $\Theta(n)$

18. De las siguientes tres afirmaciones, una es cierta y dos falsas, o bien una es falsa y dos son ciertas. Marca la que en ese sentido es diferente a las otras dos.

- (a) Ramificación y poda sirve para resolver problemas que vuelta atrás no puede.
- (b) Ramificación y poda siempre es más eficiente que vuelta atrás.
- (c) Ramificación y poda resuelve el mismo tipo de problemas que vuelta atrás.

19. 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 cierta.

- (a) La complejidad temporal de la mejor solución posible al problema es $O(n^2)$.
- (b) La complejidad temporal de la mejor solución posible al problema es $O(n!)$.
- (c) Se puede construir una solución al problema basada en el esquema de ramificación y poda, pero el uso de cotas optimistas y pesimistas no mejoraría el algoritmo resultante.

20. La relación de recurrencia $T(n) = 1 + T(n - 1)$ si $n > 1$; $T(1) = 1 \dots$

- (a) Ninguna de las otras dos opciones es cierta.
- (b) Expresa el número de llamadas recursivas que hace el algoritmo que *buscaBinaria* en el peor de los casos.
- (c) Expresa la complejidad temporal asintótica en el peor de los casos del algoritmo de búsqueda binaria.

21. Cuando un algoritmo recursivo que sigue el esquema divide y vencerás incurre en complejidades temporales prohibitivas porque se resuelven repetidamente los mismos subproblemas...

- (a) ...debemos convertirlo obligatoriamente a iterativo para evitarlo.
- (b) ...podemos guardar soluciones parciales en un almacén para evitar esa repetición, pero resolveremos indefectiblemente más problemas que si lo convertimos en iterativo.
- (c) ...podemos guardar soluciones parciales en un almacén para evitar esa repetición y puede ser que resolvamos menos problemas que si lo convertimos en iterativo.

22. Sea el vector $v[8] = \{8, 6, 4, 5, 4, 3, 2, 2\}$. Indica cuál de las siguientes opciones es cierta. (se asume la notación del lenguaje C/C++ en la que el primer elemento del vector está en la posición 0, es decir, en $v[0]$).

- (a) El vector v es un montículo máximo.
- (b) El vector v no es un montículo máximo porque el elemento $v[3]=5$ debe ser "soltado" (desplazado hacia la izquierda).
- (c) El vector v no es un montículo máximo porque el elemento $v[2]=4$ debe ser "hundido" (desplazado hacia la derecha).

23. Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n = 1 \\ \sqrt{n} + 3f(\frac{n}{3}) & n > 1 \end{cases}$$

- (a) $f(n) \in \Theta(n)$
- (b) $f(n) \in \Theta(n^2)$
- (c) $f(n) \in \Theta(\sqrt{n} \log n)$

24. Indica cuál de las siguientes afirmaciones sobre el problema de la mochila continua es cierta:

- (a) El esquema voraz podría no encontrar ninguna solución.
- (b) Un esquema voraz siempre encuentra alguna solución, aunque no sea óptima.
- (c) Se puede demostrar que un esquema voraz encuentra siempre la solución óptima.

$$f\left(\frac{n}{3}\right) + \sqrt{n} = f\left(\frac{n}{3}\right) + \sqrt{\frac{n}{3}} + \sqrt{n} =$$

$$= 2f\left(\frac{n}{3}\right) + 2\sqrt{n} \quad \begin{array}{l} \frac{n}{3} \rightarrow n = 3^k \rightarrow k = \log n \\ \text{Sea } k = \log n \end{array}$$

$$\text{general: } 2^k f\left(\frac{n}{3^k}\right) + k\sqrt{n} \quad 2^{k+1} f\left(\frac{n}{3^{k+1}}\right) + (k+1)\sqrt{n}$$

$$n = 3^k \cdot 3^k$$

25. En una carrera de coches por el desierto uno de los principales problemas es el abastecimiento de gasolina. Un participante dispone de un mapa que le indica las distancias entre las gasolineras que hay en la ruta y cree que, parándose a repostar el menor número de veces posible, podrá ganar. Para ayudarle hay que diseñar un algoritmo que le sugiera en qué gasolineras debe hacerlo. Hay que tener en cuenta que hay una única ruta posible. De entre las estrategias que se elan, ¿cuál sería la eficiente para resolver el problema?

- (a) Programación dinámica.
- (b) Algoritmo voraz.
- (c) Ramificación y poda.

26. Sea el problema de la función compuesta mínima. Si no acotamos el número máximo de operaciones posibles, un esquema de ramificación y poda:

- (a) Podría no acabar, al tener que expandir indefinidamente nuevos nodos.
- (b) Si incluimos memorización, podría no encontrar la solución óptima pero siempre acabaría.
- (c) Si incluimos memorización, siempre encuentra la solución óptima.

27. Solo una de estas tres relaciones de recurrencia es tal que $T(n) \in \Theta(n)$. ¿Cuál?

- (a) $T(n) = 1 + 2T(n/2)$ si $n > 1$; $T(1) = 1$
- (b) $T(n) = 1 + T(n/2)$ si $n > 1$; $T(1) = 1$
- (c) $T(n) = n + T(n - 1)$ si $n > 1$; $T(1) = 1$

28. Qué complejidad temporal asintótica cabe esperar de un algoritmo divide y vencerás cuya función *descomponer* produce, en tiempo constante, dos subproblemas iguales de tamaño $n - 2$ cada uno y cuya función *combinar* es lineal con n , donde n es el tamaño del problema.

- (a) $O(n \log n)$
- (b) $O(2^n)$
- (c) $O(n^2)$

$$2T\left(\frac{n}{2}\right) + 1 = 2\left(2T\left(\frac{n}{4}\right) + 1\right) + 1 = 4T\left(\frac{n}{4}\right) + 2 + 1$$

$$2^i T\left(\frac{n}{2^i}\right) + 32^i - 1$$

$$\begin{aligned} 2T(n-2) + n &= 2(2T(n-4) + n-2) + n = 4T(n-4) + 2n - 4 + n = \\ &= 4T(n-4) + 3n - 4 \end{aligned}$$

$$\text{General: } 2^i T(n-2^i) + 2^{i-1}n - 2^i \Rightarrow 2^{i-n} T(n-2^{i-n}) + 2^{i-n} \cdot n - 2^i$$

29. Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(unsigned int n) {
    if (n == 0)
        return 1;
    return f(n/2) + f(n/2) + n;
}
```

¿qué complejidades temporal y espacial cabe esperar de la función resultante?

- (a) Complejidad temporal $O(\log n)$ y espacial $O(n)$
- (b) $O(n)$, tanto temporal como espacial
- (c) Complejidad temporal $O(\log n)$ y espacial $O(1)$

30. Indica cuál de las siguientes afirmaciones es cierta:

- (a) Si un esquema de vuelta atrás encuentra la solución óptima a un problema, un esquema Voraz también la encuentra siempre.
- (b) Si un esquema de vuelta atrás encuentra la solución óptima a un problema, un esquema voraz también podría encontrarla.
- (c) Si un esquema de vuelta atrás encuentra la solución óptima a un problema, un esquema voraz no se puede plantear.

31. Dado el siguiente programa recursivo:

```
int f( int n ) { // Se asume que n >= 0
    if( n == 0 )
        return 1;
    return f(n-1) + f(n-2);
}
```

si quisieramos mejorarlo haciendo uso de la técnica de programación dinámica, ¿cuales serían las complejidades temporal y espacial más ajustadas del algoritmo resultante?

- (a) Respectivamente, $O(n)$ y $O(1)$
- (b) Ambas complejidades serían $O(1)$
- (c) Ambas complejidades serían $O(n)$

32. Se quiere desarrollar un programa que compruebe si es posible que un caballo de ajedrez, mediante una secuencia de sus movimientos permitidos, recorra todas las casillas de un tablero $N \times N$ a partir de una determinada casilla dada como entrada y sin repetir ninguna casilla. De entre las estrategias que se citan, ¿cuál sería la eficiente para resolver el problema?

- (a) Programación dinámica.
- (b) Vuelta atrás.
- (c) Algoritmo voraz.

$$\begin{aligned} f\left(\frac{n}{2}\right) + n &= 2f\left(\frac{n}{4}\right) + \frac{n}{2} + n = \\ &= 4f\left(\frac{n}{8}\right) + 2n \end{aligned}$$

$$\begin{aligned} &\text{General:} \\ &2^i f\left(\frac{n}{2^i}\right) + in \end{aligned}$$

$$n \cdot 1 + \log n$$

33. Indica cuál de las siguientes afirmaciones sobre el problema del fontanero diligente es cierta:
- (a) El esquema voraz podría no encontrar ninguna solución.
 - (b) Se puede demostrar que un esquema voraz encuentra siempre la solución óptima.
 - (c) Se puede demostrar que un esquema voraz siempre encuentra alguna solución, aunque no sea óptima.
34. ¿Cuál de los siguientes algoritmos de ordenación necesita un espacio de almacenamiento adicional al vector que se ordena con complejidad $O(n)$?
- (a) Mergesort.
 - (b) Bubblesort.
 - (c) Quicksort.
35. ¿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) La combinación de las soluciones a los subproblemas.
 - (b) La división del problema en subproblemas.
 - (c) El número de llamadas recursivas que se hacen.
36. De las siguientes tres afirmaciones, una es cierta y dos falsas, o bien una es falsa y dos son ciertas. Marca la que en ese sentido es diferente a las otras dos.
- (a) El problema de la mochila continua no se puede resolver mediante la técnica de divide y vencerás pues se podría producir un número infinito de llamadas recursivas.
 - (b) El problema de la mochila continua no se puede resolver mediante la técnica de ramificación y poda pues se podría producir un número infinito de infinitas.
 - (c) El problema de la mochila discreta no se puede resolver mediante la técnica de divide.
37. Se dispone de un vector v que almacena números enteros en orden estrictamente creciente, y se desea averiguar si existe algún elemento que cumpla $v[i] = i$. De entre las estrategias que se citan, ¿cuál sería la eficiente para resolver el problema?
- (a) Algoritmo voraz.
 - (b) Divide y vencerás.
 - (c) Programación dinámica.

38. ¿En qué caso la complejidad temporal del algoritmo de ordenación *n quicksort* es igual a la complejidad temporal del algoritmo *mergesort*?

- (a) En el caso peor de ambos.
- (b) En el caso mejor de ambos.
- (c) Tanto en el caso peor como en el caso mejor de ambos.

39. Supongamos el problema de la mochila discreta modificado de manera que cada objeto puede seleccionarse, no seleccionarse o seleccionarse solo la mitad obteniendo en este caso la mitad de su valor. ¿Qué algoritmo resulta ser más eficiente para encontrar la solución óptima?

- (a) Un algoritmo de ramificación y poda.
- (b) Un algoritmo de vuelta atrás.
- (c) Un algoritmo voraz.

40. Indica cuál de las siguientes afirmaciones es cierta:

- (a) Si un esquema de ramificación y poda encuentra la solución óptima a un problema, un esquema de vuelta atrás también la encuentra siempre.
- (b) Si un esquema de ramificación y poda encuentra la solución óptima a un problema, un esquema de vuelta atrás podría no encontrarla.
- (c) Si un esquema de ramificación y poda encuentra la solución óptima a un problema, un esquema de vuelta atrás siempre es más ineficiente.

COMPLEJIDAD: EJERCICIOS

N	ENUNCIADO	
1	¿Cuál es el objetivo de la etapa de análisis en el Diseño y Análisis de un Algoritmo?: a. Determinar el lenguaje y herramientas disponibles para su desarrollo. b. Estimar los recursos que consumirá el algoritmo una vez implementado. c. Estimar la potencia y características del equipo informático necesarios para el correcto funcionamiento del algoritmo.	
2	¿Cuál de las siguientes jerarquías de complejidades es la correcta? a. $O(1) \subset O(\lg n) \subset O(\lg \lg n) \subset \dots$ b. $\dots \subset (n!) \subset O(2^n) \subset O(n^n)$ c. $\dots \subset (2^n) \subset O(n!) \subset O(n^p)$	
3	¿Cuál de los siguientes algoritmos de ordenación tiene menor complejidad? a. Burbuja b. Inserción directa c. Mergesort	
4	¿El tiempo de ejecución de un algoritmo depende de la talla del problema? a. Sí, siempre b. No, nunca c. No necesariamente	
5	Ordena de menor a mayor las siguientes complejidades 1. $O(1)$ 2. $O(n^2)$ 3. $O(n\lg n)$ 4. $O(n!)$	a. 3,1,2 y 4 b. 1,3,2 y 4 c. 1,3,4 y 2
6	El estudio de la complejidad resulta realmente interesante para tamaños grandes de problema por varios motivos: a. Las diferencias reales en tiempo de compilación de algoritmos con diferente coste para tamaños pequeños del problema no suelen ser muy significativas. b. Las diferencias reales en tiempo de ejecución de algoritmos con diferente coste para tamaños grandes del problema no suelen ser muy significativas. c. Ninguna de las anteriores.	
7	¿Por qué se emplean funciones de coste para expresar el coste de una algoritmo? a. Para poder expresar el coste de los algoritmos con mayor exactitud b. Para que la expresión del coste del algoritmo sea válida para cualquier entrada al mismo c. Para poder expresar el coste de un algoritmo mediante una expresión matemática	
8	El caso base de una ecuación de recurrencia asociada a la complejidad temporal de un algoritmo expresa: a. El coste de dicho algoritmo en el mejor de los casos. b. El coste de dicho algoritmo en el peor de los casos. c. Ninguna de las anteriores	
9	La complejidad de la función TB es: función TB (A: vector[λ]; iz , de : N) : N var n,i:N; n=iz-de+1 opcion (n < 1) : devuelve (0) ; (n = 1) : devuelve (1) ; (n > 1) : si (A[iz] = A[de]) entonces devuelve (TB(A, iz + 1, de - 1) + 1); sino devuelve (TB(A, iz + 1, de - 1)); finsi ; fopcion fin	a. $\Theta(n)$ b. $\Theta(n \cdot \lg n)$ c. $\Theta(n^2 \cdot \lg n)$
10	Dado el polinomio $f(n) = a_m n^m + a_{m-1} n^{m-1} + \dots + a_0$, con $a_m \in \mathbb{R}^+$ entonces f pertenece al orden: a. $O(n^m)$. b. $\Omega(n^m)$. c. La dos respuestas anteriores son correctas.	
11	Si $f1(n) \in O(g1(n))$ y $f2(n) \in O(g2(n))$ entonces: a. $f1(n) \cdot f2(n) \in O(\max(g1(n), g2(n)))$ b. $f1(n) \cdot f2(n) \in O(g1(n) \cdot g2(n))$ c. Ambas son correctas	
12	Si $f1(n) \in O(g1(n))$ y $f2(n) \in O(g2(n))$ entonces: a. $f1(n) + f2(n) \in O(\max(g1(n), g2(n)))$ b. $f1(n) + f2(n) \in O(g1(n) + g2(n))$ c. Ambas son correctas	
13	Un algoritmo cuya talla es n y que tarda 40^n segundos en resolver cualquier instancia tiene una complejidad temporal: a. $\Theta(n^n)$ b. $\Theta(4^n)$ c. Ninguna de las anteriores	

14	Si dos algoritmos tienen la misma complejidad asintótica: a. No necesitan exactamente el mismo tiempo para su ejecución. b. Necesitan exactamente el mismo tiempo para su ejecución. c. Ninguna de las anteriores	
15	Los algoritmos directos de ordenación, respecto de los indirectos: a. Presentan una mayor complejidad temporal y sus tiempos de ejecución absolutos son mayores. b. Presentan una menor complejidad temporal y sus tiempos de ejecución absolutos son menores. c. Presentan una mayor complejidad temporal si bien sus tiempos de ejecución absolutos son menores.	
16	La talla o tamaño de un problema depende de: a. Conjunto de valores asociados a la entrada y salida del problema. b. Conjunto de valores asociados a la salida del problema. c. Conjunto de valores asociados a la entrada del problema.	
17	En un algoritmo recursivo, la forma de dividir el problema en subproblemas: a. Influye en la complejidad espacial del mismo. b. Influye en su complejidad temporal. c. No influye en ninguna de sus complejidades.	
18	$f(n) = 5n + 3m \cdot n + 11$ entonces $f(n)$ pertenece a: a. $O(n \cdot m)$. b. $O(n^m)$. c. Las dos son correctas	
19	El sumatorio, desde $i=1$ hasta n , de i^k pertenece a: a. $O(n^{k+1})$ b. $O(n^k)$ c. Ninguna de las anteriores	
20	La complejidad de la función A2 es: Funcion A2 (n, a: entero):entero; Var r: entero; fvar si ($a^2 > n$) devuelve 0 sino r := A2(n, 2a); opción n < a^2 : devuelve r; n $\geq a^2$: devuelve r + a; fopción fsi fin	a. $O(\sqrt{n} \cdot a)$ b. $O(\sqrt{n} / a)$ c. $O(n / \sqrt{a})$
21	Cual de las siguientes definiciones es cierta: a. Las cotas de complejidad se emplean cuando para una misma talla se obtienen diferentes complejidades dependiendo de la entrada al problema. b. Las cotas de complejidad se emplean cuando para diferentes tallas se obtienen diferentes complejidades dependiendo de la entrada al problema. c. Ninguna de las anteriores	
22	Cuando para distintas instancias de problema con el mismo tamaño no obtenemos el mismo resultado: a. No es posible calcular la complejidad a priori y debemos ejecutar el programa varias veces con la misma talla y obtener el tiempo medio para hallar la complejidad media. b. No se puede aplicar la técnica de paso de programa, ya que esta técnica es para calcular la complejidad a priori. c. Calculamos el máximo y mínimo coste que nos puede dar el algoritmo.	
23	$f(n) = 5n + 5$ ¿ $f(n)$ pertenece a $O(n)$? a. Si. El valor de c es 5 y el valor mínimo de n_0 es de 3 b. Si. El valor de c es 9 y el valor mínimo de n_0 es de 1 c. Si. El valor de c es 6 y el valor mínimo de n_0 es de 5	
24	$f(n) = 10n + 7$ ¿ $f(n)$ pertenece a $O(n^2)$? a. Si. Para c = 1 y a partir de un valor de $n_0 = 10$. b. Sí. Para cualquier valor de c positivo siempre existe un n_0 a partir del que se cumple. c. No.	
25	Si $f(n) \in \Omega(g(n))$ entonces: a. $\exists c, n_0 \in \mathbb{R}^+: f(n) \geq c \cdot g(n) \forall n \geq n_0$ b. $\exists c, n_0 \in \mathbb{R}^+: f(n) \geq c \cdot g(n) \forall n$ c. $\exists c, n_0 \in \mathbb{R}^+: f(n) \leq c \cdot g(n) \forall n \geq n_0$	
26	El coste asociado a la siguiente ecuación de recurrencia es: $f(n) = \begin{cases} 1 & n \leq 1 \\ n + f(n/2) + f(n/2) & n > 1 \end{cases}$	a. $\Theta(n \lg n^2)$ b. $\Theta(n^2 \lg n)$ c. $\Theta(n \lg n)$

14	Si dos algoritmos tienen la misma complejidad asintótica: a. No necesitan exactamente el mismo tiempo para su ejecución. b. Necesitan exactamente el mismo tiempo para su ejecución. c. Ninguna de las anteriores	
15	Los algoritmos directos de ordenación, respecto de los indirectos: a. Presentan una mayor complejidad temporal y sus tiempos de ejecución absolutos son mayores. b. Presentan una menor complejidad temporal y sus tiempos de ejecución absolutos son menores. c. Presentan una mayor complejidad temporal si bien sus tiempos de ejecución absolutos son menores.	
16	La talla o tamaño de un problema depende de: a. Conjunto de valores asociados a la entrada y salida del problema. b. Conjunto de valores asociados a la salida del problema. c. Conjunto de valores asociados a la entrada del problema.	
17	En un algoritmo recursivo, la forma de dividir el problema en subproblemas: a. Influye en la complejidad espacial del mismo. b. Influye en su complejidad temporal. c. No influye en ninguna de sus complejidades.	
18	$f(n) = 5n + 3m \cdot n + 11$ entonces $f(n)$ pertenece a: a. $O(n \cdot m)$. b. $O(n^m)$. c. Las dos son correctas	
19	El sumatorio, desde $i=1$ hasta n , de i^k pertenece a: a. $O(n^{k+1})$ b. $O(n^k)$ c. Ninguna de las anteriores	
20	La complejidad de la función A2 es: Funcion A2 (n, a: entero):entero; Var r: entero; fvar si ($a^2 > n$) devuelve 0 sino r := A2(n, 2a); opción n < a^2 : devuelve r; n $\geq a^2$: devuelve r + a; fopción fsi fin	a. $O(\sqrt{n} \cdot a)$ b. $O(\sqrt{n} / a)$ c. $O(n / \sqrt{a})$
21	Cual de las siguientes definiciones es cierta: a. Las cotas de complejidad se emplean cuando para una misma talla se obtienen diferentes complejidades dependiendo de la entrada al problema. b. Las cotas de complejidad se emplean cuando para diferentes tallas se obtienen diferentes complejidades dependiendo de la entrada al problema. c. Ninguna de las anteriores	
22	Cuando para distintas instancias de problema con el mismo tamaño no obtenemos el mismo resultado: a. No es posible calcular la complejidad a priori y debemos ejecutar el programa varias veces con la misma talla y obtener el tiempo medio para hallar la complejidad media. b. No se puede aplicar la técnica de paso de programa, ya que esta técnica es para calcular la complejidad a priori. c. Calculamos el máximo y mínimo coste que nos puede dar el algoritmo.	
23	$f(n) = 5n + 5$ ¿ $f(n)$ pertenece a $O(n)$? a. Si. El valor de c es 5 y el valor mínimo de n_0 es de 3 b. Si. El valor de c es 9 y el valor mínimo de n_0 es de 1 c. Si. El valor de c es 6 y el valor mínimo de n_0 es de 5	
24	$f(n) = 10n + 7$ ¿ $f(n)$ pertenece a $O(n^2)$? a. Si. Para c = 1 y a partir de un valor de $n_0 = 10$. b. Sí. Para cualquier valor de c positivo siempre existe un n_0 a partir del que se cumple. c. No.	
25	Si $f(n) \in \Omega(g(n))$ entonces: a. $\exists c, n_0 \in \mathbb{R}^+: f(n) \geq c \cdot g(n) \forall n \geq n_0$ b. $\exists c, n_0 \in \mathbb{R}^+: f(n) \geq c \cdot g(n) \forall n$ c. $\exists c, n_0 \in \mathbb{R}^+: f(n) \leq c \cdot g(n) \forall n \geq n_0$	
26	El coste asociado a la siguiente ecuación de recurrencia es: $f(n) = \begin{cases} 1 & n \leq 1 \\ n + f(n/2) + f(n/2) & n > 1 \end{cases}$	a. $\Theta(n \lg n^2)$ b. $\Theta(n^2 \lg n)$ c. $\Theta(n \lg n)$

DIVIDE Y VENCERAS Y PROGRAMACIÓN DINAMICA: EJERCICIOS

N	ENUNCIADO
1	¿Qué esquema algorítmico utiliza el algoritmo de ordenación Quicksort? a. Divide y Vencerás b. Programación Dinámica c. Backtracking
2	Ante un problema que presenta una solución recursiva siempre podemos aplicar: a. Divide y vencerás b. Programación dinámica c. Cualquiera de las dos anteriores
3	En cual de los siguientes casos no se puede aplicar el esquema Divide y Vencerás: a. Cuando los subproblemas son de tamaños muy diferentes b. Cuando el problema no cumple el principio de optimalidad c. Se puede aplicar en ambos casos.
4	Dado el algoritmo de búsqueda binaria, supongamos que, en vez de dividir la lista de elementos en dos mitades del mismo tamaño, la dividimos en dos partes de tamaños 1/3 y 2/3. El coste de este algoritmo: a. Es el mismo que el del original b. Es mayor que el del original c. Es menor que el del original
5	Si n es el número de elementos del vector, el coste del algoritmo Mergesort es: a. $O(n^2)$ y $\Omega(n \log n)$ b. $\Theta(n \log n)$ c. $\Theta(n^2)$
6	Un problema se puede resolver por Divide y Vencerás siempre que: a. Cumpla el principio de optimalidad b. Cumpla el teorema de reducción c. Ninguna de las anteriores
7	La serie de números de Fibonacci se define de la siguiente forma: $fib(n) = \begin{cases} 1 & n \leq 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$ Para implementar esta función podemos emplear : a. Divide y vencerás b. Programación dinámica c. Cualquiera de las dos anteriores
8	La serie de números de Fibonacci se define de la siguiente forma: $fib(n) = \begin{cases} 1 & n \leq 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$ ¿Qué implementación de entre las siguientes supone el menor coste? a. Divide y vencerás b. Programación dinámica c. Ambas tienen el mismo coste asintótico
9	El problema de la mochila, ¿puede solucionarse de forma óptima empleando la estrategia de divide y vencerás?: a. Sólo para el caso de la mochila con fraccionamiento b. Sólo para el caso de la mochila sin fraccionamiento c. Si, se puede aplicar para ambos casos.
10	Para que un problema de optimización se pueda resolver mediante PD es necesario que: a. Cumpla el principio de optimalidad b. Cumpla el teorema de reducción c. Cumpla los dos anteriores
11	Dada una solución recursiva a un problema ¿Cómo podemos evitar la resolución de los mismos subproblemas muchas veces? a. Resolver los subproblemas de mayor a menor y guardar su resultado en una tabla, inicializándola con los problemas pequeños. b. Resolver los subproblemas de menor a mayor y guardar su resultado en una tabla, inicializándola con los problemas pequeños. c. Resolver los subproblemas de mayor a menor y guardar su resultado en una tabla, inicializándola con los problemas más grandes.
12	Si aplicamos Programación Dinámica a un problema que también tiene solución por divide y vencerás podemos asegurar que... a. El coste temporal se reduce y el espacial aumenta con respecto a la solución por DyV b. El coste temporal aumenta y el espacial se reduce con respecto a la solución por DyV c. Ninguna de las anteriores.
13	¿Cuándo utilizaremos Programación Dinámica en lugar de Divide y Vencerás? a. Cuando se incrementa la eficacia b. Cuando se incrementa la eficiencia c. Cuando se reduce el coste espacial.
14	En programación dinámica, dónde almacenamos los valores de los problemas resueltos? a. En un vector unidimensional b. En un vector bidimensional c. Depende del problema

15	Supongamos el problema de la mochila resuelto mediante Programación Dinámica y particularizado para n elementos y un peso máximo trasportable de P. ¿Es necesario calcular valores para toda la matriz auxiliar para obtener el resultado? a. Si b. No c. Depende de los valores de n y P.
16	Un problema de optimización cuya solución se puede expresar mediante una secuencia de decisiones cumple el principio de optimalidad si, dada una secuencia óptima: a. Existe una subsecuencia de esa solución que corresponde a la solución óptima de su subproblema asociado b. Existe al menos una subsecuencia de esa solución que corresponde a la solución óptima de su subproblema asociado c. Cualquier subsecuencia de esa solución corresponde a la solución óptima de su subproblema asociado
17	La programación dinámica, para resolver un problema, aplica la estrategia... a. Se resuelven los problemas más pequeños y, combinando las soluciones, se obtienen las soluciones de problemas sucesivamente más grandes hasta llegar al problema original. b. Se descompone el problema a resolver en subproblemas más pequeños, que se resuelven independientemente para finalmente combinar las soluciones de los subproblemas para obtener la solución del problema original. c. Ninguna de las anteriores
18	¿Qué esquema de programación es el adecuado para resolver el problema del k-ésimo mínimo en un vector? a. Programación Dinámica b. Divide y Vencerás c. Ninguno de los dos
19	Si n es el número de elementos de un vector. La solución de menor coste al problema de encontrar su k-ésimo mínimo tiene la siguiente complejidad: a. $\Omega(n)$ y $O(n \log n)$ b. $\Omega(n)$ y $O(n^2)$ c. Ninguna de las dos
20	Si n es el número de elementos de un vector. Podemos encontrar una solución al problema de encontrar su k-ésimo que esté acotada superiormente por : a. $O(n^3)$ b. $O(n)$ c. Ninguna de las dos
21	Dada la solución recursiva al problema de encontrar el k-ésimo mínimo de un vector. Cada llamada recursiva, ¿cuántas nuevas llamadas recursivas genera? a. una o ninguna b. dos o ninguna c. una o dos
22	La solución al problema de encontrar el k-ésimo mínimo de un vector pone en práctica la siguiente estrategia: a. Ordena totalmente el vector b. Ordena parcialmente el vector c. No ordena ningún elemento del vector
23	¿Qué esquema de programación es el adecuado para resolver el problema de la búsqueda binaria? a. Programación Dinámica b. Divide y Vencerás c. Ninguno de los dos
24	Si n es el número de elementos de un vector. La solución de menor coste al problema de la búsqueda binaria tiene la siguiente complejidad: a. $\Omega(\log n)$ y $O(n \log n)$ b. $\Theta(n \log n)$ c. $\Omega(1)$ y $O(\log n)$
25	¿Con qué esquema de programación obtenemos algoritmos que calculan la distancia de edición entre dos cadenas? a. Programación Dinámica b. Divide y vencerás c. Ambos
26	Disponemos de dos cadenas de longitudes m y n. Si resolvemos el problema de la distancia de edición mediante programación dinámica, ¿De qué tamaño debemos definir la matriz que necesitaremos? a. $(m-1) \times (n-1)$ b. $m \times n$ c. $(m+1) \times (n+1)$

ALGORITMOS VORACES, VUELTA ATRÁS Y RAMIFICACIÓN Y PODA: EJERCICIOS

N	ENUNCIADO
1	El método voraz se emplea en la resolución de problemas de selección y optimización en los que se pretende encontrar: <ul style="list-style-type: none"> a. Una solución que satisfaga unas restricciones y optimice una cierta función objetivo. b. Todas las soluciones que satisfagan unas restricciones. c. La dos respuestas anteriores son correctas.
2	Voraz siempre da solución óptima: <ul style="list-style-type: none"> a. Al problema de la mochila sin fraccionamiento. b. Al problema de la mochila con fraccionamiento. c. A los dos.
3	En el método Voraz, aunque las decisiones son irreversibles, podemos asegurar que: <ul style="list-style-type: none"> a. Siempre obtendremos la solución óptima. b. Siempre obtendremos una solución factible. c. Sólo obtendremos la solución óptima para algunos problemas.
4	Dado un problema de optimización y un algoritmo Voraz que lo soluciona, ¿cuándo podemos estar seguros de que la solución obtenida será óptima?: <ul style="list-style-type: none"> a. Cuando demostremos formalmente que el criterio conduce a una solución óptima para cualquier instancia del problema. b. Voraz siempre encuentra solución óptima. c. En ambos casos. Las dos son correctas
5	Si aplicamos un algoritmo voraz que no nos garantiza la solución óptima sobre un problema entonces... <ul style="list-style-type: none"> a. Obtendremos una solución factible. b. Puede que no encuentre ninguna solución aunque ésta exista. c. Si el problema tiene solución óptima, el esquema voraz nos garantiza que la encuentra.
6	El problema de la mochila, ¿encuentra su solución óptima empleando la estrategia voraz?: <ul style="list-style-type: none"> a. Sólo para el caso de la mochila con fraccionamiento b. Sólo para el caso de la mochila sin fraccionamiento c. En cualquiera de los casos anteriores.
7	Dado un grafo G que representa las poblaciones de la provincia de Alicante de más de 20.000 habitantes junto con todas las carreteras de conexión entre ellas. Queremos obtener el recorrido que nos permita pasar por todas estas ciudades una única vez y volver al punto de origen recorriendo el mínimo número de kilómetros. Si aplicamos una estrategia voraz sobre este grafo obtendremos... <ul style="list-style-type: none"> a. Una solución factible b. La solución óptima c. Puede que no encuentre ninguna solución aunque ésta exista.
8	Al aplicar backtracking obtenemos la solución óptima a un problema: <ul style="list-style-type: none"> a. Siempre b. En algunos casos c. Sólo cuando el problema cumple el principio de Optimalidad.
9	Si aplicamos un esquema backtracking que no nos garantiza la solución óptima sobre un problema entonces... <ul style="list-style-type: none"> a. Obtendremos una solución factible. b. Puede que no encuentre ninguna solución aunque ésta exista. c. Ninguna de las anteriores.
10	Backtracking es aplicable a problemas de selección y optimización en los que: <ul style="list-style-type: none"> a. El espacio de soluciones es un conjunto infinito. b. El espacio de soluciones es un conjunto finito. c. En cualquiera de los casos
11	En un problema resuelto por backtracking, el conjunto de valores que pueden tomar las componentes de la tupla solución, ha de ser: <ul style="list-style-type: none"> a. Infinito. b. Finito c. Continuo
12	Al aplicar vuelta atrás a la solución de problemas, obtenemos algoritmos con costes computacionales: <ul style="list-style-type: none"> a. Polinómicos. b. Exponentiales. c. Los dos son correctos. Depende del problema.
13	Vuelta atrás se emplea en la resolución de problemas de optimización en los que se pretende encontrar: <ul style="list-style-type: none"> a. Una solución que satisfaga unas restricciones y optimice una cierta función objetivo. b. Todas las soluciones que satisfagan unas restricciones. c. La dos respuestas anteriores son correctas.
14	Backtracking procede a obtener la solución a un problema de optimización empleando la siguiente estrategia: <ul style="list-style-type: none"> a. Genera todas las combinaciones de la solución y selecciona la que optimiza la función objetivo. b. Genera todas las soluciones factibles y selecciona la que optimiza la función objetivo. c. Genera una solución factible empleando un criterio óptimo.

15	Batracking es una técnicas de resolución general de problemas basada en:
	a. La búsqueda sistemática de soluciones. b. La construcción directa de la solución. c. Ninguna de las anteriores
16	Batracking genera las soluciones posibles al problema:
	a. Mediante el recorrido en profundidad del árbol que representa el espacio de soluciones. b. Mediante el recorrido en anchura del árbol que representa el espacio de soluciones c. Ninguna de las anteriores
17	El problema de la mochila, ¿puede solucionarse empleando vuelta atrás?:
	a. Sólo para el caso de la mochila con fraccionamiento b. Sólo para el caso de la mochila sin fraccionamiento c. Se puede aplicar para ambos casos.
18	El problema del viajante de comercio puede resolverse correctamente empleando estos esquemas de programación:
	a. Solo backtracking b. Empleando cualquiera de estos: Voraz y backtracking. c. Sólo programación dinámica.
19	El tiempo de ejecución de un algoritmo de ramificación y poda depende de:
	a. La instancia del problema b. La función de selección de nodos para su expansión c. De ambos
20	Dado un problema resuelto mediante backtracking y mediante ramificación y poda, el coste computacional de la solución por ramificación y poda, en comparación con la de backtracking es:
	a. Igual b. Mayor c. Menor.
21	Cuál de estas afirmaciones es falsa?
	a. Batracking inspecciona todo el espacio de soluciones de un problema mientras que Ramificación y poda no. b. La complejidad en el peor caso de las soluciones Backtracking y ramificación y poda a un mismo problema es la misma. c. Para un mismo problema, ramificación y poda explora siempre un número de nodos menor o igual que backtracking.
EL PROBLEMA DE LA ASIGNACIÓN DE TURNOS.	
	Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asignen una prioridad. El número de alumnos es N. Se dispone una matriz cuadrada M con N filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre -1 y N-1) indicando dicha prioridad (un valor -1 indica que no quiere o no puede estar con la persona de la columna correspondiente, 0 indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona) . Ningún alumno puede formar grupo consigo mismo.
	Se pretende encontrar una solución para satisfacer el número máximo de alumnos según su orden de preferencia. Suponiendo que la matriz M ya existe, diseñar un algoritmo que resuelva el problema de forma óptima
22	El problema de la asignación de turnos tiene solución óptima voraz aplicando la siguiente estrategia:
	a. Seleccionamos los alumnos en orden descendente de preferencia respetando las restricciones de cabida de cada turno. b. Seleccionamos los alumnos en orden ascendente de preferencia respetando las restricciones de cabida de cada turno c. El problema no tiene solución óptima voraz
23	El problema de la asignación de turnos tiene solución óptima empleando:
	a. Batracking b. Voraz c. Ambos
24	El problema de la asignación de turnos tiene solución...
	a. Optima mediante baotracking b. Aproximada (sub-óptima) mediante voraz c. Ambas.
25	Dada la solución recursiva mediante vuelta atrás al problema de la asignación de turnos ¿cuántas nuevas llamadas recursivas genera cada llamada recursiva?
	a. una o dos b. una o ninguna c. ninguna de las anteriores
26	El problema de la asignación de turnos resuelto mediante backtracking tiene una complejidad:
	a. Exponencial b. Polinómica c. Ninguna de la dos

Un algoritmo recursivo basado en el esquema divide y vencerás ...

Seleccione una:

- a. ... será más eficiente cuanto más equitativa sea la división en subproblemas.
- b. Las demás opciones son verdaderas.
- c. ... nunca tendrá una complejidad exponencial.

La respuesta correcta es: ... será más eficiente cuanto más equitativa sea la división en subproblemas

Indicad cuál de estas tres expresiones es falsa:

Seleccione una:

- a. $\Theta(n/2) = \Theta(n)$
- b. $\Theta(n) \subseteq \Theta(n^2)$
- c. $\Theta(n) \subseteq O(n)$

La respuesta correcta es: $\Theta(n) \subseteq \Theta(n^2)$

¿Cuál de estas tres expresiones es falsa?

Seleccione una:

- a. $3n^2 + 1 \in O(n^3)$
- b. $n + n \log(n) \in \Omega(n)$
- c. $n + n \log(n) \in \Theta(n)$ ✓

Indica cuál es la complejidad en el peor caso de la función replace:

```
unsigned bound( const vector<int> &v ) {
    for( unsigned i = 0; i < v.size(); i++ )
        if( v[i] == '0' )
            return i;
    return v.size();
}

void replace( vector<int>& v, int c ) {
    for( unsigned i = 0; i < bound(v); i++)
        v[i] = c;
}
```

Seleccione una:

- a. $O(n \log n)$
- b. $O(n^2)$ ✓
- c. $O(n)$

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n) {
    if (n<=1)
        return 0;
    unsigned sum = desperdicio (n/2) + desperdicio (n/2);
    for (unsigned i=1; i<n-1; i++)
        for (unsigned j=1; j<=i; j++)
            sum+=1;
    return sum;
}
```

Seleccione una:

- a. $\Theta(n^2)$ ✓
- b. $\Theta(2^n)$
- c. $\Theta(n^2 \log n)$

Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2) + 1$; $f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a $f(n) \in \Theta(n)$
- b $f(n) \in \Theta(n^2)$
- c $f(n) \in \Theta(n \log(n))$

La respuesta correcta es: $f(n) \in \Theta(n)$

Considerad estos dos fragmentos:

```
s=0;for (i=0;i<n;i++) s+=i;
```

y

```
s=0;for (i=0;i<n;i++) if (a[i] !=0) s+=i;
```

y un array $a[i]$ de números enteros. Indicad cuál de estas tres afirmaciones es cierta:

Seleccione una:

- a. El coste temporal asintótico del primer programa en el caso peor es más alto que en el segundo.
- b. El coste temporal asintótico, tanto en el caso mejor como en el caso peor, de los dos programas es el mismo.
- c. El coste temporal asintótico del segundo programa en el caso peor es más alto que en el primero.

La respuesta correcta es: El coste temporal asintótico, tanto en el caso mejor como en el caso peor, de los dos programas es el mismo.

Indica cuál es la complejidad, en función de n , del fragmento siguiente:

```
int a = 0;
for( int i = 0; i < n; i++ )
    for( int j = i; j > 0; j /=2 )
        a += x[i][j];
```

Seleccione una:

- a $O(n \log n)$
- b $O(n)$
- c $O(n^2)$

La respuesta correcta es: $O(n \log n)$

Indica cuál es la complejidad en función de n , donde k es una constante (no depende de n), del fragmento siguiente :

```
for( int i = k; i < n - k; i++){
    A[i] = 0;
    for( int j = i - k; j < i + k; j++ )
        A[i] += B[j];
}
```

Seleccione una:

- a $O(n)$
- b $O(n \log n)$
- c $O(n^2)$

La respuesta correcta es: $O(n)$

Pertenece $3n^2 + 3$ a $O(n^3)$?

Seleccione una:

- a Sí, para $c = 1$ y $n_0 = 5$.
- b. No.
- c. Sí.

La respuesta correcta es: Sí.

La complejidad temporal en el mejor de los casos..

Seleccione una:

- a. Las demás opciones son verdaderas.
- b. ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.
- c. ... es una función de la talla que tiene que estar definida para todos los posibles valores de ésta.

La respuesta correcta es: ... es una función de la talla que tiene que estar definida para todos los posibles valores de ésta.

La versión de Quicksort que utiliza como pivote la mediana del vector ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. ... se comporta peor cuando el vector ya está ordenado.
- c. ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

La respuesta correcta es: ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n=1 \\ \sqrt{n} + 3f(n/3) & n>1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n)$
- b. $f(n) \in \Theta(n^3)$
- c. $f(n) \in \Theta(\sqrt{n} \log n)$

La respuesta correcta es: $f(n) \in \Theta(n)$

Un problema de tamaño n puede transformarse en tiempo $O(n^2)$ en nueve de tamaño $n/3$; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante.

¿Cuál de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a. $O(n^2)$
- b. $O(n^2 \log n)$
- c. $O(n \log n)$

La respuesta correcta es: $O(n^2 \log n)$

Indica cuál es la complejidad, en función de n , del siguiente fragmento de código:

```
s=0; for (i=0; i<n; i++) for (j=i; j<n; j++) s+=i+j;
```

Seleccione una:

- a. $\Theta(n^2)$ ✓
- b. $O(n^2)$ pero no $\Omega(n^2)$.
- c. $\Theta(n)$

Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n-1) + 1$, $f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a. $f(n) \in \Theta(n)$
- b. $f(n) \in \Theta(2^n)$ ✓
- c. $f(n) \in \Theta(n^2)$

Un programa con dos bucles anidados uno dentro del otro. El primero hace n iteraciones aproximadamente y el segundo la mitad, tarda un tiempo

Seleccione una:

- a. $O(n \log n)$
- b. $O(n^2)$ ✓
- c. $O(n \sqrt{n})$

Un problema de tamaño n puedo transformarlo en tiempo $O(n^2)$ en nuevo de tamaño $n/3$; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante.

¿Cuál de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a. $O(n \log n)$
- b. $O(n^2 \log n)$ ✓
- c. $O(n^2)$

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n){  
    if (n<=1)  
        return 0;  
    unsigned sum = desperdicio (n/2) + desperdicio (n/2);  
    for (unsigned i=1; i<n-1; i++)  
        for (unsigned j=1; j<-i; j++)  
            for (unsigned k=1; k<=j; k++)  
                sum+=i*j*k;  
    return sum;  
}
```

Seleccione una:

- a. $\Theta(2^n)$
- b. $\Theta(n^3 \log n)$
- c. $\Theta(n^3)$ ✓

Los algoritmos de ordenación Quicksort y Mergesort tienen en común ...

Seleccione una:

- a. ... que se ejecutan en tiempo $O(n)$.
- b. ... que ordenan el vector sin usar espacio adicional.
- c. ... que aplican la estrategia de divide y vencerás. ✓

Indica cuál es la complejidad de la función siguiente:

```
unsigned sum( const mat &A ) {    // A es una matriz cuadrada
    unsigned d = A.n_rows();
    unsigned a = 0;
    for( unsigned i = 0; i < d; i++ )
        for( unsigned j = 0; j < d; j++ )
            a += A(i,j);
    return a;
}
```

Seleccione una:

- a. $O(n^2)$
- b. $O(n)$ ✓
- c. $O(n \log n)$

Indicad cuál de estas tres expresiones es falsa:

Seleccione una:

- a. $\Theta(n/2) = \Theta(n)$
- b. $\Theta(n) \subseteq O(n)$
- c. $\Theta(n) \subseteq \Theta(n^2)$ ✓

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

- a. El árbol de cobertura de coste mínimo de un grafo conexo.
- b. El problema de la mochila discreta o sin fraccionamiento. ✓
- c. El problema de la mochila continua o con fraccionamiento.

Los algoritmos de programación dinámica hacen uso ...

Seleccione una:

- a. de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.
- b. de que se puede ahorrar cálculos guardando resultados anteriores en un almacén. ✓
- c. de una estrategia trivial consistente en examinar todas las soluciones posibles.

Cuando se calculan los coeficientes binomiales usando la recursión $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$, con $\binom{n}{0} = \binom{n}{n} = 1$, qué problema se da y cómo se puede resolver?

Seleccione una:

- a. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.
- b. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz
- c. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica. ✓

Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2) + n$; $f(1) = 1$. Indicad cuál de estas

Seleccione una:

- a. $f(n) \in \Theta(n^2)$
- b. $f(n) \in \Theta(n)$
- c. $f(n) \in \Theta(n \log(n))$ ✓

Para que la complejidad de un algoritmo presente caso mejor y peor distintos ...

Seleccione una:

- a. ... es condición necesaria y suficiente que existan instancias distintas del problema con el mismo tamaño.
- b. ... es condición necesaria que existan instancias distintas del problema con el mismo tamaño. ✓
- c. ... es condición suficiente que existan instancias distintas del problema con el mismo tamaño.

Un problema de tamaño n puede transformarse en tiempo $O(n)$ en siete de tamaño $n/7$; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante.

¿Cuál de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a. $O(n^2)$
- b. $O(n)$
- c. $O(n \log n)$ ✓

Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a. $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
- b. $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$
- c. $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$ ✓

La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

Seleccione una:

- a. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.
- b. Las demás opciones son falsas. ✓
- c. ... siempre coincidirá con la complejidad temporal de las instancias que están en el caso base del algoritmo recursivo.

Considerad la función siguiente:

```
int M( int i, int f ) {  
    if ( i == f )  
        return i;  
    else {  
        e = v[ M( i, (i+f)/2 ) ];  
        f = v[ M( (i+f)/2+1, f ) ];  
        if (e < f)  
            return e;  
        else  
            return f;  
    }  
}
```

Si la talla del problema viene dada por $n = f - i + 1$, ¿cuál es el coste temporal asintótico en el supuesto de que n sea una potencia de 2?

Seleccione una:

- a. $O(n)$ ✓
- b. $O(n^2)$.
- c. $O(n \log(n))$.

El coste temporal asintótico del fragmento

```
s=0; for(i=0;i<n;i++) for(j=i;j<n;j++) s+=i*j;
```

y el del fragmento

```
s=0; for(i=0;i<n;i++) for(j=0;j<n;j++) s+=i*i*j;
```

son ...

Seleccione una:

- a. ... iguales. ✓
- b. ... el del segundo, menor que el del primero.
- c. ... el del primero, menor que el del segundo.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n=1 \\ n^2 + 3f(n/3) & n>1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n^2)$
- b. $f(n) \in \Theta(n^2 \log n)$
- c. $f(n) \in \Theta(n)$

La versión de Quicksort que utiliza como pivote el elemento del vector que ocupa la primera posición ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. ... se comporta peor cuando el vector ya está ordenado. ✓
- c. ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

La versión de *Quicksort* que utiliza como pivote el elemento del vector que ocupa la posición central ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado. ✓
- b. ... se comporta peor cuando el vector ya está ordenado.
- c. ... no presenta casos mejor y peor distintos para instancias del mismo tamaño.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n = 1 \\ n + 3f(n/3) & n > 1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n \log n)$ ✓
- b. $f(n) \in \Theta(n^3)$
- c. $f(n) \in \Theta(n)$

Cuando se resuelve usando un algoritmo de *vuelta atrás* un problema de n decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a. $O(2^n)$ ✓
- b. $O(n!)$
- c. $O(n^2)$

Al resolver el problema del viajante de comercio mediante *vuelta atrás* y asumiendo un grafo de n vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Seleccione una:

- a. Se multiplica n por la distancia de la arista más corta que nos queda por considerar.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las n aristas más cortas.
- c. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido. ✓

Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?

Seleccione una:

- a. *Ramificación y poda*, puesto que con buenas funciones de cota es más eficiente para este problema que *vuelta atrás*.
- b. *Divide y vencerás*, puesto que la división en sublistas se podría hacer en tiempo constante.
- c. *Vuelta atrás*, para este problema no hay un esquema más eficiente. ✓

La complejidad en el mejor de los casos de un algoritmo de *ramificación y poda* ...

Seleccione una:

- a. ... es siempre exponencial con el número de decisiones a tomar.
- b. ... suele ser polinómica con el número de alternativas por cada decisión
- c. ... puede ser polinómica con el número de decisiones a tomar. ✓

La complejidad en el peor de los casos de un algoritmo de *ramificación y poda* ...

Seleccione una:

- a. ... puede ser exponencial con el número de alternativas por cada decisión.
- b. ... puede ser polinómica con el número de decisiones a tomar.
- c. ... es exponencial con el número de decisiones a tomar. ✓

La estrategia de *ramificación y poda* genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones. ✓
- b. ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- c. ... un recorrido en anchura del árbol que representa el espacio de soluciones.

¿Para qué sirven las cotas pesimistas en *ramificación y poda*?

Seleccione una:

- a. Para tener la certeza de que la cota optimista está bien calculada.
- b. Para descartar nodos basándose en la preferencia por algún otro nodo ya completado.
- c. Para descartar nodos basándose en el beneficio esperado. ✓

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- 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, nunca es así.

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
- b. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir. ✓
- c. Sí, siempre es así.

En los algoritmos de *ramificación y poda* ...

Seleccione una:

- a. El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida.
- 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 optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.

La ventaja de la estrategia *ramificación y poda* frente a *vuelta atrás* es que la primera genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.
- b. Las otras dos opciones son verdaderas. ✓
- c. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.

¿Cuál es la diferencia principal entre una solución de *vuelta atrás* y una solución de *ramificación y poda* para el problema de la mochila?

Seleccione una:

- a. El orden de exploración de las soluciones. ✓
- b. El coste asintótico en el caso peor.
- c. El hecho que la solución de *ramificación y poda* puede empezar con una solución subóptima voraz y la de *vuelta atrás* no.

Tratándose de un problema de optimización, en la lista de nodos vivos de *ramificación y poda* ...

Seleccione una:

- a. ... sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento.
- b. ... puede haber nodos que no son prometedores.
- c. Las otras dos opciones son ciertas. ✓

Cuando resolvemos un problema mediante un esquema de *ramificación y poda* ...

Seleccione una:

- a. ... los valores entre los cuales se elige en cada una de las decisiones tienen que formar un conjunto finito. ✓
- b. ... las decisiones sólo pueden ser binarias.
- c. ... los valores entre los cuales se elige en cada una de las decisiones pueden formar un conjunto infinito.

La estrategia de ramificación y poda necesita cotas pesimistas ...

Seleccione una:

- a. ... para decidir el orden de visita de los nodos del árbol de soluciones.
- b. ... sólo si se usa para resolver problemas de optimización. ✓
- c. ... para determinar si una solución es factible.

El uso de funciones de cota en ramificación y poda ...

Seleccione una:

- a. ... transforma en polinómicas complejidades que antes eran exponenciales.
- b. ... puede reducir el número de instancias del problema que pertenecen al caso peor. ✓
- c. ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.

En la estrategia de *ramificación y poda* ...

Seleccione una:

- a. ... cada nodo tiene su propia cota pesimista y también su propia cota optimista. ✓
- b. ... cada nodo tiene su propia cota pesimista, la cota optimista sin embargo, es común para todos los nodos.
- c. ... cada nodo tiene su propia cota optimista, la cota pesimista sin embargo, es común para todos los nodos.

Si para resolver un mismo problema usamos un algoritmo de *vuelta atrás* y lo modificamos mínimamente para convertirlo en un algoritmo de *ramificación y poda*, ¿qué cambiamos realmente?

Seleccione una:

- a. La comprobación de las soluciones factibles: en *ramificación y poda* no es necesario puesto que sólo genera nodos factibles. ✗
- b. Cambiamos la función que damos a la cota pesimista.
- c. Aprovechamos mejor las cotas optimistas.

Si para resolver un mismo problema usamos un algoritmo de *ramificación y poda* y lo modificamos mínimamente para convertirlo en un algoritmo de *vuelta atrás*, ¿qué cambiamos realmente?

Seleccione una:

- a. Cambiamos la función que damos a la cota pesimista.
- b. Provocamos que las cotas optimistas pierdan eficacia.
- c. Será necesario comprobar si las soluciones son factibles o no puesto que *ramificación y poda* sólo genera nodos factibles.

En el esquema de *vuelta atrás*, los mecanismos de poda basados en la mejor solución hasta el momento ...

Seleccione una:

- a. Las dos anteriores son verdaderas.
- b. ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- c. ... pueden eliminar soluciones parciales que son factibles. ✓

En ausencia de cotas optimistas y pesimistas, la estrategia de *vuelta atrás* ...

Seleccione una:

- a. ... no se puede usar para resolver problemas de optimización.
- b. ... debe recorrer siempre todo el árbol.
- c. ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles. ✓

La estrategia de *vuelta atrás* es aplicable a problemas de selección y optimización en los que:

Seleccione una:

- a. El espacio de soluciones puede ser tanto finito como infinito pero en este último caso debe ser al menos numerable.
- b. El espacio de soluciones es un conjunto infinito.
- c. El espacio de soluciones es un conjunto finito. ✓

Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila:

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- c. El valor óptimo de la mochila continua correspondiente.

Decid cuál de estas tres no sirve como cota optimista para obtener el valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- b. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos. ✓
- c. El valor de la mochila continua correspondiente.

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor de la mochila continua correspondiente.
- c. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

Seleccione una:

- a. podemos aplicar el esquema *vuelta atrás* siempre que se trate de un conjunto infinito numerable.
- b. es probable que a través de *programación dinámica* se obtenga un algoritmo eficaz que lo solucione.
- c. una estrategia voraz puede ser la única alternativa. ✓

Dado un problema de optimización cualquiera, ¿la estrategia de *vuelta atrás* garantiza la solución óptima?

Seleccione una:

- a. Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.
- b. Sí, puesto que ese método analiza todas las posibilidades.
- c. 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.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

Seleccione una:

- a. Sería una cota pesimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible. ✓
- b. Ninguna de las otras dos opciones.
- c. Sería una cota optimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. Como también se conocen las coordenadas geográficas de cada ciudad se quiere usar la distancia geográfica (en línea recta) entre cada par de ciudades para como cota para limitar la búsqueda en un algoritmo de *vuelta atrás*.

¿Qué tipo de cota sería?

Seleccione una:

- a. No se trataría de ninguna poda puesto que es posible que esa heurística no encuentre una solución factible.
- b. Una cota pesimista.
- c. Una cota optimista. ✓

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, \$-1\$) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. 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.

Seleccione una:

- a. El nuevo algoritmo solo será más rápido para algunas instancias del problema. ✓
- b. Esta estrategia no asegura que se obtenga el camino mas corto.
- c. El nuevo algoritmo siempre sea más rápido.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino según su distancia geográfica.

Este algoritmo voraz ¿serviría como cota pesimista?

Seleccione una:

- a. No, ya que no asegura que se encuentre una solución factible. ✓
- b. Sí, puesto que la distancia geográfica asegura que otra solución mejor no es posible.
- c. No, ya que en algunos casos puede dar distancias menores que la óptima.

El problema de cortar un tubo de longitud n en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...

Seleccione una:

- a. ... no se puede resolver usando un algoritmo de *vuelta atrás*.
- b. ... se puede resolver mediante un algoritmo de *vuelta atrás* pero existe una solución asintóticamente mucho más eficiente. ✓
- c. ... se debe resolver mediante un algoritmo de *vuelta atrás*, dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo.

Al resolver el problema del viajante de comercio mediante *vuelta atrás* y asumiendo un grafo de n vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Seleccione una:

- a. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las n aristas más cortas.
- c. Se multiplica n por la distancia de la arista más corta que nos queda por considerar.

Al resolver el problema del viajante de comercio mediante *vuelta atrás*, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

Seleccione una:

- 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. ✓

Di cuál de estas tres soluciones a problemas de optimización no comporta, en el peor caso, tener que considerar $O(n!)$ posibilidades.

Seleccione una:

- a. La solución de *vuelta atrás* al problema del viajante de comercio (*travelling salesman problem*), o sea, el de encontrar un ciclo hamiltoniano de coste mínimo en un grafo conexo de n vértices donde cada arista tiene un coste asignado. X
- b. La solución de *ramificación y poda* al problema de la asignación de n tareas a n trabajadores de forma que cada trabajador hace exactamente una tarea y cada tarea es asignada a un trabajador exactamente, de forma que la suma de los costes de las tareas es mínimo. X
- c. La solución al problema de buscar un árbol que cubre todos los vértices de un grafo de n vértices de forma que el coste es mínimo (*minimum spanning tree*).

Un algoritmo recursivo basado en el esquema divide y vencerás ...

Seleccione una:

- a. ... será más eficiente cuanto más equitativa sea la división en subproblemas.
- b. Las demás opciones son verdaderas.
- c. ... nunca tendrá una complejidad exponencial.

La respuesta correcta es: ... será más eficiente cuanto más equitativa sea la división en subproblemas

Indicad cuál de estas tres expresiones es falsa:

Seleccione una:

- a. $\Theta(n/2) = \Theta(n)$
- b. $\Theta(n) \subseteq \Theta(n^2)$
- c. $\Theta(n) \subseteq O(n)$

La respuesta correcta es: $\Theta(n) \subseteq \Theta(n^2)$

¿Cuál de estas tres expresiones es falsa?

Seleccione una:

- a. $3n^2 + 1 \in O(n^3)$
- b. $n + n \log(n) \in \Omega(n)$
- c. $n + n \log(n) \in \Theta(n)$ ✓

Indica cuál es la complejidad en el peor caso de la función replace:

```
unsigned bound( const vector<int> &v ) {
    for( unsigned i = 0; i < v.size(); i++ )
        if( v[i] == '0' )
            return i;
    return v.size();
}

void replace( vector<int>& v, int c ) {
    for( unsigned i = 0; i < bound(v); i++)
        v[i] = c;
}
```

Seleccione una:

- a. $O(n \log n)$
- b. $O(n^2)$ ✓
- c. $O(n)$

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n) {
    if (n<=1)
        return 0;
    unsigned sum = desperdicio (n/2) + desperdicio (n/2);
    for (unsigned i=1; i<n-1; i++)
        for (unsigned j=1; j<=i; j++)
            sum+=1;
    return sum;
}
```

Seleccione una:

- a. $\Theta(n^2)$ ✓
- b. $\Theta(2^n)$
- c. $\Theta(n^2 \log n)$

Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2) + 1$; $f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a $f(n) \in \Theta(n)$
- b $f(n) \in \Theta(n^2)$
- c $f(n) \in \Theta(n \log(n))$

La respuesta correcta es: $f(n) \in \Theta(n)$

Considerad estos dos fragmentos:

```
s=0;for (i=0;i<n;i++) s+=i;
```

y

```
s=0;for (i=0;i<n;i++) if (a[i] !=0) s+=i;
```

y un array $a[i]$ de números enteros. Indicad cuál de estas tres afirmaciones es cierta:

Seleccione una:

- a. El coste temporal asintótico del primer programa en el caso peor es más alto que en el segundo.
- b. El coste temporal asintótico, tanto en el caso mejor como en el caso peor, de los dos programas es el mismo.
- c. El coste temporal asintótico del segundo programa en el caso peor es más alto que en el primero.

La respuesta correcta es: El coste temporal asintótico, tanto en el caso mejor como en el caso peor, de los dos programas es el mismo.

Indica cuál es la complejidad, en función de n , del fragmento siguiente:

```
int a = 0;
for( int i = 0; i < n; i++ )
    for( int j = i; j > 0; j /=2 )
        a += x[i][j];
```

Seleccione una:

- a $O(n \log n)$
- b $O(n)$
- c $O(n^2)$

La respuesta correcta es: $O(n \log n)$

Indica cuál es la complejidad en función de n , donde k es una constante (no depende de n), del fragmento siguiente :

```
for( int i = k; i < n - k; i++){
    A[i] = 0;
    for( int j = i - k; j < i + k; j++ )
        A[i] += B[j];
}
```

Seleccione una:

- a. $O(n)$
- b. $O(n \log n)$
- c. $O(n^2)$

La respuesta correcta es: $O(n)$

Pertenece $3n^2 + 3$ a $O(n^3)$?

Seleccione una:

- a. Sílo para $c = 1$ y $n_0 = 5$.
- b. No.
- c. Sí.

La respuesta correcta es: Sí.

La complejidad temporal en el mejor de los casos..

Seleccione una:

- a. Las demás opciones son verdaderas.
- b. ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.
- c. ... es una función de la talla que tiene que estar definida para todos los posibles valores de ésta.

La respuesta correcta es: ... es una función de la talla que tiene que estar definida para todos los posibles valores de ésta.

La versión de Quicksort que utiliza como pivote la mediana del vector ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. ... se comporta peor cuando el vector ya está ordenado.
- c. ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

La respuesta correcta es: ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n=1 \\ \sqrt{n} + 3f(n/3) & n>1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n)$
- b. $f(n) \in \Theta(n^3)$
- c. $f(n) \in \Theta(\sqrt{n} \log n)$

La respuesta correcta es: $f(n) \in \Theta(n)$

Un problema de tamaño n puede transformarse en tiempo $O(n^2)$ en nueve de tamaño $n/3$; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante.

¿Cuál de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a. $O(n^2)$
- b. $O(n^2 \log n)$
- c. $O(n \log n)$

La respuesta correcta es: $O(n^2 \log n)$

Indica cuál es la complejidad, en función de n , del siguiente fragmento de código:

```
s=0; for (i=0; i<n; i++) for (j=i; j<n; j++) s+=i+j;
```

Seleccione una:

- a. $\Theta(n^2)$ ✓
- b. $O(n^2)$ pero no $\Omega(n^2)$.
- c. $\Theta(n)$

Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n-1) + 1$, $f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a. $f(n) \in \Theta(n)$
- b. $f(n) \in \Theta(2^n)$ ✓
- c. $f(n) \in \Theta(n^2)$

Un programa con dos bucles anidados uno dentro del otro. El primero hace n iteraciones aproximadamente y el segundo la mitad, tarda un tiempo

Seleccione una:

- a. $O(n \log n)$
- b. $O(n^2)$ ✓
- c. $O(n \sqrt{n})$

Un problema de tamaño n puedo transformarlo en tiempo $O(n^2)$ en nuevo de tamaño $n/3$; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante.

¿Cuál de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a. $O(n \log n)$
- b. $O(n^2 \log n)$ ✓
- c. $O(n^2)$

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n){  
    if (n<=1)  
        return 0;  
    unsigned sum = desperdicio (n/2) + desperdicio (n/2);  
    for (unsigned i=1; i<n-1; i++)  
        for (unsigned j=1; j<-i; j++)  
            for (unsigned k=1; k<=j; k++)  
                sum+=i*j*k;  
    return sum;  
}
```

Seleccione una:

- a. $\Theta(2^n)$
- b. $\Theta(n^3 \log n)$
- c. $\Theta(n^3)$ ✓

Los algoritmos de ordenación Quicksort y Mergesort tienen en común ...

Seleccione una:

- a. ... que se ejecutan en tiempo $O(n)$.
- b. ... que ordenan el vector sin usar espacio adicional.
- c. ... que aplican la estrategia de divide y vencerás. ✓

Indica cuál es la complejidad de la función siguiente:

```
unsigned sum( const mat &A ) {    // A es una matriz cuadrada
    unsigned d = A.n_rows();
    unsigned a = 0;
    for( unsigned i = 0; i < d; i++ )
        for( unsigned j = 0; j < d; j++ )
            a += A(i,j);
    return a;
}
```

Seleccione una:

- a. $O(n^2)$
- b. $O(n)$ ✓
- c. $O(n \log n)$

Indicad cuál de estas tres expresiones es falsa:

Seleccione una:

- a. $\Theta(n/2) = \Theta(n)$
- b. $\Theta(n) \subseteq O(n)$
- c. $\Theta(n) \subseteq \Theta(n^2)$ ✓

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

- a. El árbol de cobertura de coste mínimo de un grafo conexo.
- b. El problema de la mochila discreta o sin fraccionamiento. ✓
- c. El problema de la mochila continua o con fraccionamiento.

Los algoritmos de programación dinámica hacen uso ...

Seleccione una:

- a. de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.
- b. de que se puede ahorrar cálculos guardando resultados anteriores en un almacén. ✓
- c. de una estrategia trivial consistente en examinar todas las soluciones posibles.

Cuando se calculan los coeficientes binomiales usando la recursión $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$, con $\binom{n}{0} = \binom{n}{n} = 1$, qué problema se da y cómo se puede resolver?

Seleccione una:

- a. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.
- b. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz
- c. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica. ✓

Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2) + n$; $f(1) = 1$. Indicad cuál de estas

Seleccione una:

- a. $f(n) \in \Theta(n^2)$
- b. $f(n) \in \Theta(n)$
- c. $f(n) \in \Theta(n \log(n))$ ✓

Para que la complejidad de un algoritmo presente caso mejor y peor distintos ...

Seleccione una:

- a. ... es condición necesaria y suficiente que existan instancias distintas del problema con el mismo tamaño.
- b. ... es condición necesaria que existan instancias distintas del problema con el mismo tamaño. ✓
- c. ... es condición suficiente que existan instancias distintas del problema con el mismo tamaño.

Un problema de tamaño n puede transformarse en tiempo $O(n)$ en siete de tamaño $n/7$; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante.

¿Cuál de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a. $O(n^2)$
- b. $O(n)$
- c. $O(n \log n)$ ✓

Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a. $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
- b. $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$
- c. $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$ ✓

La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

Seleccione una:

- a. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.
- b. Las demás opciones son falsas. ✓
- c. ... siempre coincidirá con la complejidad temporal de las instancias que están en el caso base del algoritmo recursivo.

Considerad la función siguiente:

```
int M( int i, int f ) {
    if ( i == f )
        return i;
    else {
        e = v[ M( i, (i+f)/2 ) ];
        f = v[ M( (i+f)/2+1, f ) ];
        if (e < f)
            return e;
        else
            return f;
    }
}
```

Si la talla del problema viene dada por $n = f - i + 1$, ¿cuál es el coste temporal asintótico en el supuesto de que n sea una potencia de 2?

Seleccione una:

- a. $O(n)$. ✓
- b. $O(n^2)$.
- c. $O(n \log(n))$.

El coste temporal asintótico del fragmento

```
s=0; for(i=0;i<n;i++) for(j=i;j<n;j++) s+=i*j;
```

y el del fragmento

```
s=0; for(i=0;i<n;i++) for(j=0;j<n;j++) s+=i*i*j;
```

son ...

Seleccione una:

- a. ... iguales. ✓
- b. ... el del segundo, menor que el del primero.
- c. ... el del primero, menor que el del segundo.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n=1 \\ n^2 + 3f(n/3) & n>1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n^2)$
- b. $f(n) \in \Theta(n^2 \log n)$
- c. $f(n) \in \Theta(n)$

La versión de Quicksort que utiliza como pivote el elemento del vector que ocupa la primera posición ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. ... se comporta peor cuando el vector ya está ordenado. ✓
- c. ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

La versión de *Quicksort* que utiliza como pivote el elemento del vector que ocupa la posición central ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado. ✓
- b. ... se comporta peor cuando el vector ya está ordenado.
- c. ... no presenta casos mejor y peor distintos para instancias del mismo tamaño.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n = 1 \\ n + 3f(n/3) & n > 1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n \log n)$ ✓
- b. $f(n) \in \Theta(n^3)$
- c. $f(n) \in \Theta(n)$

Cuando se resuelve usando un algoritmo de *vuelta atrás* un problema de n decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a. $O(2^n)$ ✓
- b. $O(n!)$
- c. $O(n^2)$

Al resolver el problema del viajante de comercio mediante *vuelta atrás* y asumiendo un grafo de n vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Seleccione una:

- a. Se multiplica n por la distancia de la arista más corta que nos queda por considerar.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las n aristas más cortas.
- c. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido. ✓

Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?

Seleccione una:

- a. *Ramificación y poda*, puesto que con buenas funciones de cota es más eficiente para este problema que *vuelta atrás*.
- b. *Divide y vencerás*, puesto que la división en sublistas se podría hacer en tiempo constante.
- c. *Vuelta atrás*, para este problema no hay un esquema más eficiente. ✓

La complejidad en el mejor de los casos de un algoritmo de *ramificación y poda* ...

Seleccione una:

- a. ... es siempre exponencial con el número de decisiones a tomar.
- b. ... suele ser polinómica con el número de alternativas por cada decisión
- c. ... puede ser polinómica con el número de decisiones a tomar. ✓

La complejidad en el peor de los casos de un algoritmo de *ramificación y poda* ...

Seleccione una:

- a. ... puede ser exponencial con el número de alternativas por cada decisión.
- b. ... puede ser polinómica con el número de decisiones a tomar.
- c. ... es exponencial con el número de decisiones a tomar. ✓

La estrategia de *ramificación y poda* genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones. ✓
- b. ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- c. ... un recorrido en anchura del árbol que representa el espacio de soluciones.

¿Para qué sirven las cotas pesimistas en *ramificación y poda*?

Seleccione una:

- a. Para tener la certeza de que la cota optimista está bien calculada.
- b. Para descartar nodos basándose en la preferencia por algún otro nodo ya completado.
- c. Para descartar nodos basándose en el beneficio esperado. ✓

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- 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, nunca es así.

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
- b. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir. ✓
- c. Sí, siempre es así.

En los algoritmos de *ramificación y poda* ...

Seleccione una:

- a. El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida.
- 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 optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.

La ventaja de la estrategia *ramificación y poda* frente a *vuelta atrás* es que la primera genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.
- b. Las otras dos opciones son verdaderas. ✓
- c. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.

¿Cuál es la diferencia principal entre una solución de *vuelta atrás* y una solución de *ramificación y poda* para el problema de la mochila?

Seleccione una:

- a. El orden de exploración de las soluciones. ✓
- b. El coste asintótico en el caso peor.
- c. El hecho que la solución de *ramificación y poda* puede empezar con una solución subóptima voraz y la de *vuelta atrás* no.

Tratándose de un problema de optimización, en la lista de nodos vivos de *ramificación y poda* ...

Seleccione una:

- a. ... sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento.
- b. ... puede haber nodos que no son prometedores.
- c. Las otras dos opciones son ciertas. ✓

Cuando resolvemos un problema mediante un esquema de *ramificación y poda* ...

Seleccione una:

- a. ... los valores entre los cuales se elige en cada una de las decisiones tienen que formar un conjunto finito. ✓
- b. ... las decisiones sólo pueden ser binarias.
- c. ... los valores entre los cuales se elige en cada una de las decisiones pueden formar un conjunto infinito.

La estrategia de ramificación y poda necesita cotas pesimistas ...

Seleccione una:

- a. ... para decidir el orden de visita de los nodos del árbol de soluciones.
- b. ... sólo si se usa para resolver problemas de optimización. ✓
- c. ... para determinar si una solución es factible.

El uso de funciones de cota en ramificación y poda ...

Seleccione una:

- a. ... transforma en polinómicas complejidades que antes eran exponenciales.
- b. ... puede reducir el número de instancias del problema que pertenecen al caso peor. ✓
- c. ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.

En la estrategia de *ramificación y poda* ...

Seleccione una:

- a. ... cada nodo tiene su propia cota pesimista y también su propia cota optimista. ✓
- b. ... cada nodo tiene su propia cota pesimista, la cota optimista sin embargo, es común para todos los nodos.
- c. ... cada nodo tiene su propia cota optimista, la cota pesimista sin embargo, es común para todos los nodos.

Si para resolver un mismo problema usamos un algoritmo de *vuelta atrás* y lo modificamos mínimamente para convertirlo en un algoritmo de *ramificación y poda*, ¿qué cambiamos realmente?

Seleccione una:

- a. La comprobación de las soluciones factibles: en *ramificación y poda* no es necesario puesto que sólo genera nodos factibles. ✗
- b. Cambiamos la función que damos a la cota pesimista.
- c. Aprovechamos mejor las cotas optimistas.

Si para resolver un mismo problema usamos un algoritmo de *ramificación y poda* y lo modificamos mínimamente para convertirlo en un algoritmo de *vuelta atrás*, ¿qué cambiamos realmente?

Seleccione una:

- a. Cambiamos la función que damos a la cota pesimista.
- b. Provocamos que las cotas optimistas pierdan eficacia.
- c. Será necesario comprobar si las soluciones son factibles o no puesto que *ramificación y poda* sólo genera nodos factibles.

En el esquema de *vuelta atrás*, los mecanismos de poda basados en la mejor solución hasta el momento ...

Seleccione una:

- a. Las dos anteriores son verdaderas.
- b. ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- c. ... pueden eliminar soluciones parciales que son factibles. ✓

En ausencia de cotas optimistas y pesimistas, la estrategia de *vuelta atrás* ...

Seleccione una:

- a. ... no se puede usar para resolver problemas de optimización.
- b. ... debe recorrer siempre todo el árbol.
- c. ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles. ✓

La estrategia de *vuelta atrás* es aplicable a problemas de selección y optimización en los que:

Seleccione una:

- a. El espacio de soluciones puede ser tanto finito como infinito pero en este último caso debe ser al menos numerable.
- b. El espacio de soluciones es un conjunto infinito.
- c. El espacio de soluciones es un conjunto finito. ✓

Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila:

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- c. El valor óptimo de la mochila continua correspondiente.

Decid cuál de estas tres no sirve como cota optimista para obtener el valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- b. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos. ✓
- c. El valor de la mochila continua correspondiente.

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor de la mochila continua correspondiente.
- c. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

Seleccione una:

- a. podemos aplicar el esquema *vuelta atrás* siempre que se trate de un conjunto infinito numerable.
- b. es probable que a través de *programación dinámica* se obtenga un algoritmo eficaz que lo solucione.
- c. una estrategia voraz puede ser la única alternativa. ✓

Dado un problema de optimización cualquiera, ¿la estrategia de *vuelta atrás* garantiza la solución óptima?

Seleccione una:

- a. Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.
- b. Sí, puesto que ese método analiza todas las posibilidades.
- c. 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.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

Seleccione una:

- a. Sería una cota pesimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible. ✓
- b. Ninguna de las otras dos opciones.
- c. Sería una cota optimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. Como también se conocen las coordenadas geográficas de cada ciudad se quiere usar la distancia geográfica (en línea recta) entre cada par de ciudades para como cota para limitar la búsqueda en un algoritmo de *vuelta atrás*.

¿Qué tipo de cota sería?

Seleccione una:

- a. No se trataría de ninguna poda puesto que es posible que esa heurística no encuentre una solución factible.
- b. Una cota pesimista.
- c. Una cota optimista. ✓

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, \$-1\$) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. 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.

Seleccione una:

- a. El nuevo algoritmo solo será más rápido para algunas instancias del problema. ✓
- b. Esta estrategia no asegura que se obtenga el camino mas corto.
- c. El nuevo algoritmo siempre sea más rápido.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino según su distancia geográfica.

Este algoritmo voraz ¿serviría como cota pesimista?

Seleccione una:

- a. No, ya que no asegura que se encuentre una solución factible. ✓
- b. Sí, puesto que la distancia geográfica asegura que otra solución mejor no es posible.
- c. No, ya que en algunos casos puede dar distancias menores que la óptima.

El problema de cortar un tubo de longitud n en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...

Seleccione una:

- a. ... no se puede resolver usando un algoritmo de *vuelta atrás*.
- b. ... se puede resolver mediante un algoritmo de *vuelta atrás* pero existe una solución asintóticamente mucho más eficiente. ✓
- c. ... se debe resolver mediante un algoritmo de *vuelta atrás*, dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo.

Al resolver el problema del viajante de comercio mediante *vuelta atrás* y asumiendo un grafo de n vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Seleccione una:

- a. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las n aristas más cortas.
- c. Se multiplica n por la distancia de la arista más corta que nos queda por considerar.

Al resolver el problema del viajante de comercio mediante *vuelta atrás*, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

Seleccione una:

- 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. ✓

Di cuál de estas tres soluciones a problemas de optimización no comporta, en el peor caso, tener que considerar $O(n!)$ posibilidades.

Seleccione una:

- a. La solución de *vuelta atrás* al problema del viajante de comercio (*travelling salesman problem*), o sea, el de encontrar un ciclo hamiltoniano de coste mínimo en un grafo conexo de n vértices donde cada arista tiene un coste asignado. X
- b. La solución de *ramificación y poda* al problema de la asignación de n tareas a n trabajadores de forma que cada trabajador hace exactamente una tarea y cada tarea es asignada a un trabajador exactamente, de forma que la suma de los costes de las tareas es mínimo. X
- c. La solución al problema de buscar un árbol que cubre todos los vértices de un grafo de n vértices de forma que el coste es mínimo (*minimum spanning tree*).

Preguntas:

1. Decid cuál de estas tres estrategias proveería la cota pesimista más ajustada al valor óptimo de la mochila discreta:

- (a) Completar las decisiones restantes basándose en la mejor solución voraz que pueda encontrarse para los restantes objetos y espacio disponible de la mochila.
- (b) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- (c) Asumir que ya no se van a coger más objetos.



2. ¿Cuál de estos problemas tiene una solución eficiente utilizando programación dinámica?

- (a) El problema del cambio.
- (b) La mochila discreta con pesos y valores reales positivos.
- (c) El problema de la asignación de tareas.



$O(n^k)$

3. 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) $O(\log n)$
- (b) $O(n^2)$
- (c) $O(n)$

$$f(n) = 1 + f(n-2)$$

$$f(4) = 1 + f(2) + 1 + f(0) = 3 + f(0)$$

$$f(n) = i + f(n-2i)$$



$$n-2i = 0 \Rightarrow n = 2i$$

$$i = \frac{n}{2}$$

4. 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) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
- (c) 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.



5. ¿En un esquema de RyP, ¿podrían coincidir los valores obtenidos por las cotas pesimista y optimista de un nodo cualquiera?

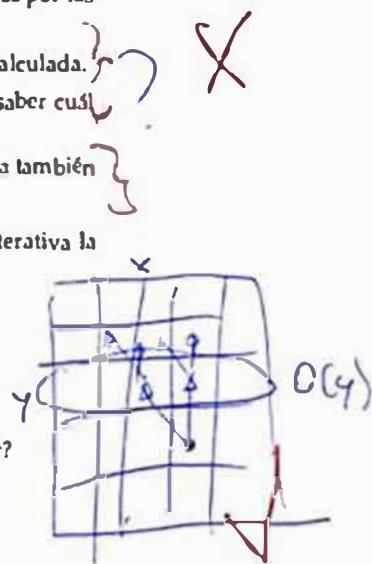
- (a) No, en tal caso una de las cotas (o ambas) estaría mal calculada.
- (b) Si, pero habría que seguir completando el nodo para saber cuál es la mejor solución que puede obtenerse de él.
- (c) Si, en tal caso el valor obtenido por las cotas coincidiría también con el mejor valor que puede obtenerse de ese nodo.

6. Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f( int y, int x){    // suponemos y >= x
    if (x==0 || y==x) return 1;
    return f(y-1, x-1) + f(y-1, x);
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

- (a) $O(x \cdot y)$
- (b) $O(y)$
- (c) $O(x)$



7. ¿Cuál de las siguientes estrategias de búsqueda es más apropiada en un esquema de vuelta atrás?

- (a) Explorar primero los nodos con mejor valor hasta el momento en la función que se pretende optimizar.
- (b) Ninguna de las otras dos estrategias es compatible con el esquema de vuelta atrás.
- (c) Explorar primero los nodos con mejor cota optimista.

8. 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) $\Omega(n^2) \subset \Omega(n^3)$ F
- (b) $\Theta(n^2) \subset \Theta(n^3)$ F
- (c) $O(n^2) \subset O(n^3)$ V



9. De las siguientes afirmaciones marca la que es verdadera.

- (a) En un esquema de vuelta atrás, las cotas pesimistas no tienen sentido si lo que se pretende es obtener todas las soluciones factibles.
- (b) El esquema de vuelta atrás no es compatible con el uso conjunto de cotas pesimistas y optimistas.
- (c) Las cotas pesimistas no son compatibles con un esquema de vuelta atrás.



10. El algoritmo de ordenación Quicksort divide el problema en dos subproblemas. ¿Cuál es la complejidad temporal asintótica de realizar esa división?

- (a) $\Theta(\log n)$
- (b) $\Theta(n \log n)$
- (c) Ninguna de las otras dos opciones es correcta.

$\Theta(n)$ ✓

11. Se desea ordenar una lista enlazada de n elementos adaptando el 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) $\Theta(n^2)$
- (b) Ninguna de las otras dos opciones es cierta.
- (c) $\Theta(n \log n)$

✓

12. El esquema de vuelta atrás ...

- (a) Garantiza que encuentra la solución óptima a cualquier problema de selección discreta.
- (b) Se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado. No, solo con x finito
- (c) Las otras dos opciones son ambas verdaderas. No

✓ principio de optimidad

13. Queremos resolver mediante vuelta atrás el problema de las 8 reinas (colocar 8 reinas en un tablero de ajedrez de manera que no se maten mutuamente). Una buena cota optimista permitiría:

- (a) Muy probablemente, explorar menos nodos.
- (b) Muy probablemente, resolver el problema de forma más rápida.
- (c) No es aplicable este tipo de podas a este problema.

X

14. Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es par y que se utilizará la siguiente recurrencia: $\text{pot}(x, n) = \text{pot}(x, n/2) + \text{pot}(x, n/2)$; ¿Qué estrategia resulta ser más eficiente en cuanto al coste temporal?

- (a) En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.
- (b) Un algoritmo recursivo con memoización.
- (c) Divide y vencerás.

✓

porque resuelto
no de los lados
ya no tienes que
resolver el otro

15. Dada la siguiente función (donde $\max(a, b) \in \Theta(1)$):

```
float exa(vector<float>&v, vector<int>&p, int P, int i)
{
    float a, b;
    if (i >= 0) {
        if (p[i] <= P)
            a = v[i] + exa(v, p, P - p[i], i - 1);
        else a = 0;
        b = exa(v, p, P, i - 1);
    }
    return max(a, b);
}
```

Marcad la opción correcta.

- (a) La complejidad temporal en el mejor de los casos es $\Omega(n^2)$
- (b) La complejidad temporal en el peor de los casos es $O(n^2)$
- (c) La complejidad temporal en el peor de los casos es $O(2^n)$

16. Si $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$ entonces ...

- (a) ... $f(n) \in \Theta(g(n))$
- (b) ... $f(n) \in \Omega(g(n))$
- (c) ... $f(n) \in O(g(n))$

+ orden u

de arriba

$$\text{ej: } g(n) = n, f(n) = n^2 \checkmark$$

17. ¿Qué nos proporciona la media aritmética entre el coste temporal asintótico (o complejidad temporal) en el peor caso y el coste temporal asintótico en el mejor caso?

- (a) En general, nada de interés.
- (b) El coste temporal asintótico en el caso medio. NO
- (c) $T'(n) \in \Theta(n)$ y $T'(n) \in \Omega(n^2)$



18. 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) $O(n^2) \subset O(2^{\log_2 n})$ F
- (b) $\Omega(n^2) \subset \Omega(n)$ V
- (c) ~~$n \log_2 n \in \Theta(n - \sqrt{n \log_2 n})$~~ ?

19. 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) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
- (c) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.

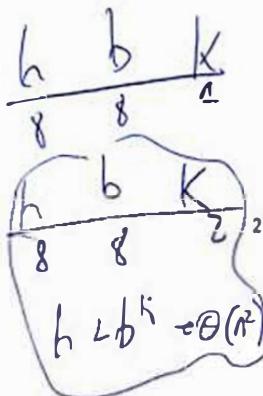


se expone como
un árbol



$$\frac{h \ b \ K}{8 \ 8 \ 3}$$

20. Sea la siguiente relación de recurrencia



$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 8T\left(\frac{n}{3}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si $T(n) \in \Theta(n^2)$, ¿en cuál de estos tres casos nos podemos encontrar?

(a) $g(n) = n^3$

(b) $g(n) = n$

(c) $g(n) = n^2$



21. En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

(a) podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.

(b) una estrategia voraz puede ser la única alternativa.

(c) es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione.



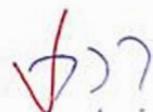
22. En un algoritmo de optimización resuelto mediante ramificación y poda ¿Podría encontrarse la solución óptima sin haber alcanzado nunca un nodo hoja?



(a) Sí, pero esto solo podría ocurrir si se hace uso de cotas pesimistas.

(b) Si, esto puede ocurrir incluso si no se hace uso de cotas pesimistas.

(c) No, los nodos hojas son los nodos completados y por lo tanto hay que visitar al menos uno de ellos para almacenarlo como la mejor solución hasta el momento.



23. Si $f \notin O(g_1)$ y $f \in O(g_2)$ entonces siempre se cumplirá:

(a) $f \notin O(\max(g_1, g_2))$ NO

(b) $f \in \Omega(g_1 + g_2)$

(c) $f \in \Omega(\min(g_1, g_2))$



24. 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, o óptimas.

(c) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.



25. ¿Qué complejidad se obtiene a partir de la relación de recurrencia $T(n) = 9T(n/3) + n^3$ con $T(1) = O(1)$?

(a) $O(n \log n)$

(b) $O(n^3 \log n)$

(c) $O(n^3)$

$$\frac{h}{9} \quad \frac{b}{3} \quad \frac{K}{3}$$

$$h \leq b^K \quad \Theta(n^3)$$

26. Se pretende resolver el problema del viajante de comercio (*travelling salesman problem*) mediante el esquema de vuelta atrás. ¿Cuál de los siguientes valores se espera que se comporte mejor para decidir si un nodo es prometedor?

- (a) El valor que se obtiene de multiplicar k por el peso de la arista más corta de entre las restantes, donde k es el número de ciudades que quedan por visitar.
- (b) La suma de los pesos de las aristas que completan la solución paso a paso visitando el vértice más cercano al último visitado.
- (c) La suma de los pesos de las k aristas restantes más cortas, donde k es el número de ciudades que quedan por visitar. *Optimista - la mejor posibl*

Problema de
viajante de
comercio

777)

$m \setminus n$	10	9	8	7	4
10	7				
9	7				
8	7				
7	7				
4	7				

27. Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f( int m, int n ) {
    if( m == n ) return 1;
    return m * f(m-1, n) + n;
}
```

$\rightarrow M = n$ tanto temporal como espacial

¿Qué complejidades temporal y espacial cabe esperar de la función resultante?

- (a) Ninguna de las otras dos opciones es correcta.
 - (b) $O(n - m)$, tanto espacial como temporal.
 - (c) $O(m - n)$, tanto espacial como temporal.
28. Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente pequeño?
- (a) Que se podría explorar más nodos de los necesarios.
 - (b) Que se podría explorar menos nodos de los necesarios.
 - (c) Que se podría podar el nodo que conduce a la solución óptima.

M
temporal?

29. ¿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) $\Theta(n \cdot m)$
- (b) $\Theta(n)$
- (c) $\Theta(n^2)$

✓

30. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...

- (a) ... no se puede usar para resolver problemas de optimización.
- (b) ... debe recorrer siempre todo el árbol.
- (c) ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.

31. Si el coste temporal de un algoritmo es $T(n)$, ¿cuál de las siguientes situaciones es imposible?

- (a) $T(n) \in O(n)$ y $T(n) \in \Theta(n)$ ✓
- (b) $T(n) \in \Omega(n)$ y $T(n) \in \Theta(n^2)$ ✓
- (c) $T(n) \in \Theta(n)$ y $T(n) \in \Omega(n^2)$ F

32. 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. NO
- (b) Las otras dos opciones son ambas falsas. NO
- (c) Siempre es mayor o igual que la mejor solución posible alcanzada.

33. Dada la siguiente función:

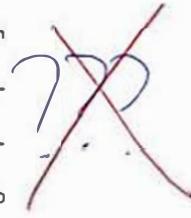
```
int exa (vector <int> v){  
    int j, i=1, n=v.size();  
  
    if (n>1) do{  
        int x = v[1];  
        for (j=1; j > 0 and v[j-1] > x; j--)  
            v[j]=v[j-1];  
        v[j]=x;  
        i++;  
    } while (i<n);  
    return 0;  
}
```

Marcad la opción correcta.

- (a) La complejidad temporal en el mejor de los casos es $\Omega(n)$ → este algoritmo ordena
- (b) La complejidad temporal exacta es $\Theta(n^2)$
- (c) La complejidad temporal en el mejor de los casos es $\Omega(1)$

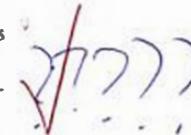
34. Con respecto a la complejidad espacial de los algoritmos de ordenación Quicksort, Heapsort y Mergesort ...

- (a) Mergesort y Heapsort tienen complejidad espacial lineal con el tamaño del vector a ordenar, la de Quicksort es constante.
- (b) Mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante.
- (c) Las complejidades espaciales de todos ellos son lineales con el tamaño del vector a ordenar.



35. ¿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) Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.
- (c) Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.



36. Un algoritmo recursivo basado en el esquema divide y vencerás...

- (a) ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a. Falso
- (b) Las otras dos opciones son ambas verdaderas. Falso
- (c) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.



37. ¿Qué tienen en común los algoritmos de ordenación Quicksort y Mergesort.

- (a) El número de llamadas recursivas que hacen en el mejor de los casos.
- (b) La complejidad temporal de la combinación de las soluciones parciales.
- (c) La complejidad temporal de la división en subproblemas.



38. Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones recurrentes expresa mejor su complejidad temporal para el caso general, siendo n el número de discos?

- (a) $T(n) = T(n-1) + n$
- (b) $T(n) = 2T(n-1) + 1$
- (c) $T(n) = 2T(n-1) + n$



$$f(n) = \sum_{i=2}^n i + f(n-i) = n + n-1 + \dots + 2 + f(n-2) = n + n-1 + \dots + 2 + f(1)$$

$$f(n) = \sum_{i=1}^{n-1} i + f(n-i) = n + n-1 + \dots + 2 + f(1)$$

$$f(n) = \sum_{i=1}^{n-1} i + f(n-i) = n + n-1 + \dots + 2 + f(1)$$

$$f(n) = \sum_{i=1}^{n-1} i + f(n-i) = n + n-1 + \dots + 2 + f(1)$$



39. La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde $g(n)$ es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Dicuál de las siguientes afirmaciones es cierta:

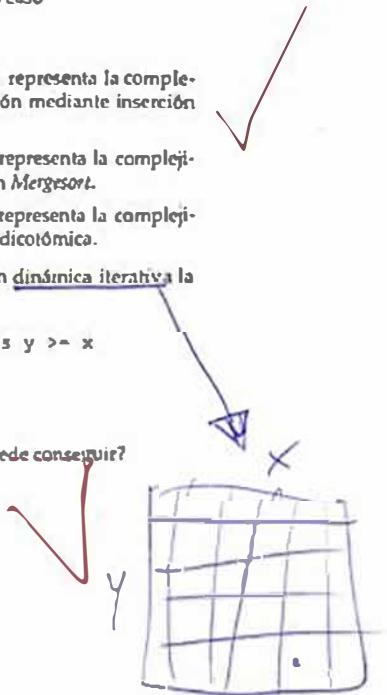
- (a) Si $g(n) \in \Theta(n^2)$ la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación mediante inserción binaria.
- (b) Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación Mergesort.
- (c) Si $g(n) \in \Theta(1)$ la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

40. Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f( int y, int x){    // suponemos y >= x
    if (x==0 || y==x) return 1;
    return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad temporal que se puede conseguir?

- (a) $O(x \cdot y)$
- (b) $O(y)$
- (c) $O(x)$



30. En ausencia de colas optimistas y pesimistas, la estrategia de vuelta atrás...

- (a) ... no se puede usar para resolver problemas de optimización.
- (b) ... debe recorrer siempre todo el árbol.
- (c) ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.

31. Si el coste temporal de un algoritmo es $T(n)$, ¿cuál de las siguientes situaciones es imposible?

- (a) $T(n) \in O(n)$ y $T(n) \in \Theta(n)$ ✓
- (b) $T(n) \in \Omega(n)$ y $T(n) \in \Theta(n^2)$ ✓
- (c) $T(n) \in \Theta(n)$ y $T(n) \in \Omega(n^2)$ F

32. Dado un problema de minimización, resuelto mediante un esquema de ramificación y poda, ¿qué propiedad cumple una cola optimista?

- (a) Asegura un ahorro en la comprobación de todas las soluciones factibles. No
- (b) Las otras dos opciones son ambas falsas.
- (c) Siempre es mayor o igual que la mejor solución posible alcanzada. ND

33. 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[i]=x;  
        i++;  
    } while (i<n);  
    return 0;  
}
```

Marcad la opción correcta.

- (a) La complejidad temporal en el mejor de los casos es $\Omega(n)$ → este algoritmo ordena
- (b) La complejidad temporal exacta es $\Theta(n^2)$
- (c) La complejidad temporal en el mejor de los casos es $\Omega(1)$

- 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 discret 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 recursiva 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 padece 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 va 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 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 aciertas.
- 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. ...es una función del tamaño o talla del problema que tiene que estar definida para todos los posibles valores de ésta.

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 encuentra ninguna solución.
- b. Que se considere la decisión ya tomada anteriormente respecto a la selección de un elemento a la vista de la decisión que se debe a tomar en el instante actual.
- c. Que la solución no sea la óptima.

13. Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

- a. Programación dinámica.
- b. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
- c. El método voraz.

14. Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si $T(n) \in O(n^2)$, ¿en cuál de estos tres casos nos podemos encontrar?

- a. $g(n) = n$
- b. $g(n) = n^2$
- c. $g(n) = 1$

15. Un algoritmo recursivo basado en el esquema divide y vencerás...

- a. ...nunca tendrá una complejidad exponencial.
- b. ...será más eficiente cuanto más equitativa sea la división en subproblemas.
- c. Las dos anteriores son ciertas.

16. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

- a. ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- b. Las otras dos opciones son ciertas.
- c. ...pueden eliminar soluciones parciales que son factibles.

17. Uno de estos tres problemas no tiene una solución eficiente que siga el esquema de programación dinámica

- a. El problema de la mochila discreta.
- b. El problema de las torres de Hanoi.

c. El problema de cortar un tubo de longitud n en segmentos de longitud entera entre 1 y n de manera que se maximice el precio de acuerdo con una tabla que da el precio para cada longitud.

18. El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

- a. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
- b. ... una cota superior para el valor óptimo.
- c. ... una cota inferior para el valor óptimo que a veces puede ser igual a este.

19. Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

- a. El valor de la mochila continua correspondiente.
- b. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido
- c. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

20. Un problema de tamaño n puede transformarse en tiempo $O(n^2)$ en otro de tamaño $n-1$. Por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante. ¿cuál de estas clases de coste temporal asintótico es la más ajustada?

- a. $O(2^n)$
- b. $O(n^3)$
- c. $O(n^2)$

21. El siguiente programa resuelve el problema de cortar un tubo de longitud n en segmentos de longitud entera entre 1 y n de manera que se maximice el precio de acuerdo con una tabla que da el precio para cada longitud, pero falta un trozo. ¿Qué debería ir en lugar de XXXXXXXX?

```

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(XXXXXXXX));
    }
    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 costa pesimista para el problema de encontrar el camino más corto entre dos ciudades (se supone que el grafo es conexo).

- a. Calcular la distancia geométrica (en línea recta entre la ciudad origen y destino).
- b. Para todas las ciudades que son alcanzables en un paso desde la ciudad inicial, sumar la distancia a dicha ciudad y la distancia geométrica hasta la ciudad destino.
- c. Calcular la distancia recorrida moviéndose al azar por el grafo hasta llegar (por azar) a la ciudad destino.

25. Si para resolver un mismo problema usamos un algoritmo de vuelta atrás y lo modificamos mínimamente para convertirlo en un algoritmo de ramificación y poda, ¿qué cambiamos realmente?

- a. Cambiamos la función que damos a la cota pesimista.
- b. El algoritmo puede aprovechar mejor las cotas optimistas.**
- c. La comprobación de las soluciones factibles: en ramificación y poda no es necesario puesto que sólo genera nodos factibles.

26. Dadas las siguientes funciones:

```
// Precondición: { 0 <= i < v.size(); i < j <= v.size() }
unsigned f( const vector<unsigned>&v, unsigned i, unsigned j ) {
    if( i == j+1 )
        return v[i];
    unsigned sum = 0;
    for( unsigned k = 0; k < j - i; k++ )
        sum += f( v, i, i+k+1 ) + f( v, i+k+1, j );
    return sum;
}

unsigned g( const vector<unsigned>&v ) {
    return f( v, v.begin(), v.end() );
}
```

Se quiere reducir la complejidad temporal de la función g usando programación dinámica iterativa. ¿cuál sería la complejidad espacial?

- a. cúbica.
- b. cuadrática.**
- c. exponencial.

27. En una cuadrícula se quiere dibujar el contorno de un cuadrado de n casillas de lado. ¿cuál será la complejidad temporal del mejor algoritmo que pueda existir?

- a. $O(n^2)$
- b. $O(n)$**
- c. $O(\sqrt{n})$

28. En los algoritmos de ramificación y poda, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (se entiende que ambas cotas se aplican sobre el mismo nodo)

- a. No, nunca es así.
- b. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.**
- c. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.

29. Cuando se resuelve el problema de la mochila discreta usando la estrategia de vuelta atrás, ¿puede ocurrir que se tarde menos en encontrar la solución óptima si se prueba a 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 tue. 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 $f(n) \in \Omega(n^3)$, ¿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.optimistic_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.

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. $\Omega(n^2)$ y $O(n^2)$
- c. $O(n)$

7. Uno de estos tres problemas no tiene una solución trivial y eficiente que siga el esquema voraz.

- a. **El problema del cambio.**
- b. El problema de la mochila discreta sin limitación en la carga máxima de la mochila.
- c. El problema de la mochila continua.

8. Estudiad la relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT\left(\frac{n}{q}\right) + g(n) & \text{en otro caso} \end{cases}$$

(donde p y q son enteros mayores que 1). Di cuál de los siguientes esquemas algorítmicos produce de manera natural relaciones de recurrencia así.

- a. **Divide y vencerás**
- b. Ramificación y poda
- c. Programación dinámica

Preg 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 un atabla que da el precio para cada longitud, pero falta un trozo. ¿Qué debería ir en lugar de XXXXXXXX?

```
void fill (price m[ ]) {
for (index i=0; i<=n; i++) m[ i ]=-1;
}

price cutrod (length n, price m[ ], price p [ ]) {
price q;
if (m[n]>=0) return m[n];
}
```

```

if (n==0) q=0;
else {
    q=-1;
    for (index i=1; i<=n; i++)
        q=max (q, p[i] + cutrod (XXXXXXX));
}
m[n]=q;
return q;
}

a) n-m[n], m, p
b) n-i, m, p
c) n, m[n] - 1, p

```

Preg 12. Sea $g(n) = \sum_{i=0}^K a_i n^i$. Dí cuál de las siguientes afirmaciones es falsa:

- a) Las otras dos afirmaciones son ambas falsas.
- b) $g(n) \in \Theta(n^k)$
- c) $g(n) \in \Omega(n^k)$

Preg 13. Si $\lim_{n \rightarrow \infty} f(n) / g(n) = 0$ entonces...

- a) ... $f(n) \in \Theta(g(n))$
- b) ... $f(n) \in O(g(n))$
- c) ... $g(n) \in O(f(n))$

Preg 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

Preg 17. En un algoritmo de ramificación y poda, si la lista de nodos vivos no está ordenada de forma apropiada ...

- a) ... podría ocurrir que se exploren nodos de forma innecesaria.
- b) ... podría ocurrir que se descarten nodos factibles.
- c) ... podría ocurrir que se pade el nodo que conduce a la solución óptima.

Preg 18. Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si $T(n) \in O(n)$, ¿en cuál de estos tres casos nos podemos encontrar?

- a) $g(n) = n^2$
- b) $g(n) = n$
- c) $g(n) = 1$

Preg 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.

Preg 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.

Preg 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) $O(n \log n)$

Preg 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

Preg 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 $O(2^n)$

Preg 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.

Preg 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.

Preg 32. La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde $g(n)$ es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

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.

Preg 33. ¿Cuál es la definición correcta de $O(f)$?

- a) $O(f) = \{g : N \rightarrow R^+ | \forall c \in R, \forall n_0 \in N, \forall n \geq n_0, f(n) \leq cg(n)\}$
- b) $O(f) = \{g : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n \geq n_0, g(n) \leq cg(n)\}$
- c) $O(f) = \{g : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n \geq n_0, f(n) \leq cg(n)\}$

Preg 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

Preg 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.**

Preg 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 $\Omega(n^2)$.

Preg 7. La complejidad temporal de la solución de vuelta atrás al problema de la mochila discreta es ...

- a) ... cuadrática en el caso peor.
- b) ... exponencial en cualquier caso.
- c) **... exponencial en el caso peor.**

Preg 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) Segundo el tipo de cota, las pesimistas puede que no descarten ningún nodo pero el uso de cotas optimistas garantiza la reducción del espacio de búsqueda.
- b) No, las cotas tanto optimistas como pesimistas garantizan la reducción del espacio de soluciones y por lo tanto la eficiencia del algoritmo.
- c) **Sí, puesto que es posible que a pesar de utilizar dichas cotas no se descarte ningún nodo.**

Preg 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.

Preg 18. Si $\lim (f(n) / n^2) = k$, 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) $f(n) \in \Theta(n^2)$

Preg 17. ¿Cuál es la definición correcta de $\Omega(g)$?

- a) $\Omega(g) = \{f : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n \geq n_0, g(n) \geq cf(n)\}$
- b) $\Omega(g) = \{f : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n \geq n_0, f(n) \geq cg(n)\}$
- c) $\Omega(g) = \{f : N \rightarrow R^+ | \forall c \in R, \exists n_0 \in N, \forall n \geq n_0, g(n) \geq cf(n)\}$

Preg 14. ¿Cuál es la definición correcta de $\Omega(f)$?

- a) $\Omega(f) = \{g : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n \geq n_0, g(n) \geq cf(n)\}$
- b) $\Omega(f) = \{g : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n \geq n_0, f(n) \geq cg(n)\}$
- c) $\Omega(f) = \{g : N \rightarrow R^+ | \forall c \in R, \exists n_0 \in N, \forall n \geq n_0, f(n) \geq cg(n)\}$

Preg 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) $n.optimistic_b() \leq pb$

- b) $n.\text{pessimistic_b}() \leq pb$
- c) $n.\text{optimistic_b}() \geq pb$

Preg 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) $\Theta(\log^2(n)) = \Theta(\log^3(n))$

Preg 17. La función y 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.

Preg 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.

Preg 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) $O(1)$

Preg 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) $n^2 \log(n)$

Preg 24. Los algoritmos voraces...

- a) ... se basan en el hecho de que una organización apropiada de los datos permite descartar la mitad de ellos en cada paso.

- b) ... se basan en el hecho de que se puede ahorrar esfuerzo guardando los resultados de cálculos anteriores en una tabla.
- c) ... se basan en la idea de que la solución óptima se puede construir añadiendo repetidamente el mejor elemento disponible.

Preg 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.

Preg 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.

Preg 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) $O(2^{\log_2(n)}) \subseteq O(n^2) \subset O(2!)$

Preg 33. Si $f \in \Theta(g1)$ y $f \in (g2)$ entonces

- a) $f2 \in \Theta(g1 * g2)$
- b) Las otras dos opciones son ambas ciertas.
- c) $f \in \Theta(\max(g1, g2))$

Preg 34. La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde $g(n)$ es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es falsa:

- a) Si $g(n) \in \Theta(1)$ la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.
- b) Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación quicksort.

- c) Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación mergesort.

Preg 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.**

Preg. 39. Dada la relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT\left(\frac{n}{a}\right) + g(n) & \text{en otro caso} \end{cases}$$

(donde p y a son enteros mayores que 1 y $g(n) = n^k$), ¿qué tiene que ocurrir para que se cumpla $T(n) \in \Theta(n^k \log_a(n))$

- a) $p > a^k$
- b) **$p = a^k$**
- c) $p > a^k$

Preg 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.

Preg 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

Preg 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.

Preg 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) $\Theta(\log_2(n)) = \Theta(\log_3(n))$

Preg 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.

Preg 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) $O(4^{\log_2(n)}) \subset O(n) \subset O(2^n)$

Preg 15. Dada la relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT\left(\frac{n}{a}\right) + g(n) & \text{en otro caso} \end{cases}$$

(donde p y a son enteros mayores que 1 y $g(n) = n^k$), ¿qué tiene que ocurrir para que se cumpla $T(n) \in \Theta(n^k)$?

- a) $p > a^k$
- b) $p < a^k$
- c) $p = a^k$

Preg 22. Si $\lim_{n \rightarrow \infty} (f(n)/n^2) = k$, y $k \neq 0$, ¿cuál de estas tres afirmaciones es cierta?

- a) $f(n) \in \Omega(n^3)$
- b) $f(n) \in \Theta(n^2)$
- c) $f(n) \in \Theta(n^3)$

Preg 24. Si $f \in \Theta(g_1)$ y $f \in \Theta(g_2)$ entonces

- a) $f \in \Theta(g_1 * g_2)$
- b) $f \notin \Theta(\max(g_1, g_2))$
- c) $f \in \Theta(g_1 + g_2)$

32. Supongamos el algoritmo de ordenación Mergesort modificado de manera que, en lugar de dividir el vector en dos partes, se divide en tres. Posteriormente se combinan las soluciones parciales. ¿Cuál sería la complejidad temporal asintótica de la combinación de las soluciones parciales?

- a) $\Theta(n)$
- b) $\Theta(\log_3 n)$
- c) Ninguna de las otras opciones es cierta

25. ¿Cuál es la complejidad temporal de la siguiente función?

```
unsigned examen (unsigned n){  
    unsigned i=n, k=0;  
    while (i>0){  
        unsigned j=i;  
        do{  
            j = j * 2;  
            k = k +1;  
        } while (j <= n);  
        i = i / 2;  
    }  
    return k;  
}
```

- a. $\Theta(\log^2 n)$
- b. $\Theta(\log n)$
- c. $\Theta(n)$

En el problema del coloreado de grafos(minimo de numeros de colores necesarios para colorear todos los vértices de un grafo de manera que no queden dos adyacentes con el mismo color) resulto mediante ramificación y poda, un 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) ...solo 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 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) $(\log 2(n)) = \Theta(\log 3(n))$

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

26. De las siguientes expresiones, o bien dos son verdaderas y una falsa es falsa, o bien dos son falsa y una verdadera. Marca la que (en este sentido) es distinta a las otras dos.

- a. $\Theta(f) = O(f) \cap \Omega(f)$
- b. $\Omega(f) = \Theta(f) \cap O(f)$
- c. $O(f) = \Omega(f) \cap \Theta(f)$

30. La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde $g(n)$ es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

a) Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación quicksort.

(b) Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal en el caso peor del algoritmo de ordenación quicksort.

(c) Si $g(n) \in \Theta(1)$ la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación mergesort.

35. ¿Cuál de estas afirmaciones es cierta?

(a) La ventaja de la solución de programación dinámica iterativa al problema de la mochila discreta es que nunca se realizan cálculos innecesarios.

b) La memoización evita que un algoritmo recursivo ingenuo resuelva repetidamente el mismo problema.

(c) Los algoritmos iterativos de programación dinámica utilizan memoización para evitar resolver de nuevo los mismos subproblemas que se vuelven a presentar.

1. Dado el problema del laberinto con tres movimientos, ¿se puede aplicar un esquema de programación dinámica para obtener un camino de sali-da?

- (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.

1. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...

- a)... debe recorrer siempre todo el árbol.
- b)... no se puede usar para resolver problemas de optimización.
- c)... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.

3. Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:

- (a) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- b)El valor de la mochila continua correspondiente .
- c)El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

**4. El coste temporal asintótico de insertar un elemento en un vector orde-nado de forma que continue ordenado
es...**

- a) $\Omega(n^2)$
- b) O(n)
- c) O(log n)

5. ¿En ramificación y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?

- (a) Sí, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista.
- b)No, la cota optimista sólo se utiliza para determinar si una n-tupla es prometedora.
- c) Sí, aunque no es una garantía de que sea una buena estrategia de búsqueda.

6. ¿Qué estrategia de búsqueda es a priori más apropiada en un esquema de vuelta atrás?

- a) Explorar primero los nodos que están más completados.
- b) En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.
- c) Explorar primero los nodos con mejor cota optimista.

7. Sea la siguiente relación de recurrencia

- a) Las otras dos opciones son ambas ciertas.
- b) $g(n) = \log n$
- c) $g(n)=\sqrt{n}$

8. Si $f \in \Omega(g_1)$ y $f \in \Omega(g_2)$ entonces

- a. $f \in \Omega(g_1 + g_2)$
- b. $f \notin \Omega(\min(g_1, g_2))$
- c. $f \in \Omega(g_1 \cdot g_2)$

9. Dada la siguiente función:

```
int exa (string & cad, int pri, int ult){  
    if (pri>=ult)  
        return 1;  
    else  
        if (cad[pri]==cad[ult])  
            return exa(cad, pri+1, ult-1);  
        else  
            return 0;  
}
```

¿Cuál es su complejidad temporal asintótica?

- (a) $O(n^2)$
- b) $O(n)$
- (c) $O(\log n)$

10. El esquema de vuelta atrás ...

- a) Se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado.
- b) Las otras dos opciones son ambas verdaderas.
- c) Garantiza que encuentra la solución óptima a cualquier problema de selección discreta.

11. En el esquema de ramificación y poda, ¿qué estructura es la más adecuada si queremos realizar una exploración por niveles?

- (a) Cola de prioridad
- (b) Pila
- c) Cola**

12. ¿Qué nos proporciona la media entre el coste temporal asintótico (o complejidad temporal) en el peor caso y el coste temporal asintótico en el mejor caso?

- a) Nada de interés.**
- (b) El coste temporal promedio.
- (c) El coste temporal asintótico en el caso medio.

14. ¿Qué complejidad se obtiene a partir de la relación de recurrencia $T(n) = 8T(n/2) + n^3$ con $T(1) = O(1)$?

- a) $O(n^3 \log n)$**
- b) $O(n^3)$
- c) $O(n \log n)$

13. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial (1,1) hasta la casilla (n, m) y para ello se aplica un esquema de programación dinámica. En cuanto a la complejidad temporal, ¿cuál es la mejora de la versión recursiva con memoización frente a la recursiva ingenua que se obtiene a partir del esquema divide y vencerás?

- a) La mejora no está garantizada puesto que la versión recursiva con memoización podría ser peor que la obtenida a partir del esquema divide y vencerás.
- d) De una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización.**
- e) De una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización.

15. Dada la siguiente función:

```
int exa (vector <int>& v) {
    int j, i=1, n=v.size();
    if (n>1) do{
        int x = v[i];
        for (j=i; j >0 and v[j-1] >x; j--)
            v[j]=v[j-1];
        v[j]=x;
        i++;
    }
```

```

        } while (i<n);
    return 0;
}

```

Marcad la opción correcta.

- a) La complejidad temporal exacta es $\Theta(n^2)$
- b) La complejidad temporal en el mejor de los casos es $\Omega(n)$
- c) La complejidad temporal en el mejor de los casos es $\Omega(1)$

16. El esquema voraz ...

- a) Las otras dos opciones son ambas falsas.
- b) Garantiza encontrar una solución a cualquier problema, aunque puede que no sea óptima.
- c) Puede que no encuentre una solución pero si lo hace se garantiza que es óptima.

17. Cuando la descomposición de un problema da lugar a subproblemas de tamaño similar al original, muchos de los cuales se repiten, ¿qué esquema es a priori más apropiado?

- a) Ramificación y poda.
- b) Divide y vencerás.
- c) Programación dinámica.

18. Dada la siguiente función(donde $\max(a, b) \in \Theta(1)$):

```

float exa(vector<float>&v, vector<int>&p, int P, int i)
{
    float a, b;
    if (i>=0){
        if (p[i] <= P)
            a= v[i]+exa(v,p,P-p[i],i-1);
        else a= 0;
        b= exa(v,p,P,i-1);
        return max(a,b);
    }
    return 0;
}

```

Marcad la opción correcta

- a) La complejidad temporal en el peor de los casos es $O(n^2)$
- b) La complejidad temporal en el peor de los casos es $O(2^n)$
- c) la complejidad temporal en el mejor de los casos es $\Omega(n^2)$

19. ¿Cuál sería la complejidad temporal de la siguiente función tras aplicar programación dinámica?

```
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.

La complejidad temporal en el peor de los casos es $O(n^2)$

La complejidad temporal en el peor de los casos es $O(2^n)$

La complejidad temporal en el mejor de los casos es $11(n^2)$

**20. Dado el problema del laberinto con tres movimientos, se pretende cono-
cer la longitud del camino de salida más corto. Para ello se aplica el es-
quema 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 desca-
rta 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 se-
lección no lo es.**
- c) Que en realidad no es un algoritmo voraz pues las decisiones que se toman no
deberían reconsiderarse

**21. Si el coste temporal de un algoritmo es $T(n)$, ¿cuál de las siguientes situa-
ciones es imposible**

- a) $T(n) \in e(n)$ y $T(n) \in 11(n^2)$**
- b) $T(n) \in 0(n)$ y $T(n) \in 9(n)$
- c) $T(n) \in 11(n)$ y $T(n) \in e(n^2)$

**22. Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es
par y que se utilizará la siguiente recurrencia: $\text{pot}(x, n) = \text{pot}(x, n/2) * \text{pot}(x, n/2)$;
¿Qué esquema resulta ser más eficiente en cuanto al coste temporal?**

- a) En este caso tanto programación dinámica como divide y vencerás resultan ser
equivalentes en cuanto a la complejidad temporal.

- b) Programación dinámica.
- c) Divide y vencerás.

23. De las siguientes afirmaciones marca la que es verdadera.

- a) El esquema de vuelta atrás no es compatible con el uso conjunto de cotas pesimistas y optimistas.
- b) Las cotas pesimistas no son compatibles con un esquema de vuelta atrás.
- c) En un esquema de vuelta atrás, las cotas pesimistas no tienen sentido si lo que se pretende es obtener todas las soluciones factibles.

24. Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿qué propiedad cumple una cota optimista?

- a) Asegura un ahorro en la comprobación de todas las soluciones factibles.
- b) Siempre es mayor o igual que la mejor solución posible alcanzada.
- c) Las otras dos opciones son ambas falsas.

25. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial (1, 1) hasta la casilla (n, m) y para ello se aplica el esquema programación dinámica para obtener un algoritmo lo más eficiente posible en cuanto a complejidad temporal y espacial. ¿Cuáles serían ambas complejidades?

- (a) Temporal $\Theta(n \times m)$ y espacial $\Theta(n \times m)$
- b) Temporal $\Theta(n \times nz)$ y espacial $\Theta(\min\{n, m\})$
- (c) Temporal $\Theta(mg\max\{n, m\})$ y espacial $\Theta(\max\{n, m\})$

26. ¿Qué se deduce de $f(n)$ y $g(n)$ si se cumple $\lim_{n \rightarrow \infty} (f(n)/g(n)) = k$, con $k \neq 0$?

- a) $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
- b) $f(n) \in O(g(n))$ pero $g(n) \notin O(f(n))$
- c) $g(n) \in O(f(n))$ pero $f(n) \notin O(g(n))$

27. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?

- a) Vuelta atrás, es el esquema más eficiente para este problema.
- b) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
- c) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.

28. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial (1, 1) hasta la casilla (n, m) y para ello se aplica un esquema de divide y vencerás. ¿Cuál sería la recurrencia apropiada para el caso general?

- (a) $nc(n, m) = nc(n - 1, m) + nc(n, m - 1) + nc(n - 1, m - 1)$
- (b) $nc(n, m) = nc(n - 1, m) * nc(n, m - 1) * nc(n - 1, m - 1)$
- (c) Ninguna de las otras dos recurrencias se corresponde con un esquema de divide y vencerás.

29. Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?

- a) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.
- b) un vector de booleanos.
- c) un par de enteros que indiquen los cortes realizados y el valor acumulado.

30. De las siguientes expresiones, o bien dos son verdaderas y una es falsa, o bien dos son falsas y una es verdadera. Marca la que (en este sentido) es distinta a las otras dos.

- (a) $\Theta(n) \subset \Theta(n^2)$
- (b) $n + n \log n \in \Omega(n)$
- (c) $O(2^{\log n}) \subset O(n^2)$

31. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

- a) Las otras dos opciones son ambas verdaderas.
- b) ... pueden eliminar vectores que representan posibles soluciones factibles.
- c) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.

32. En el problema del viajante de comercio (travelling salesman problem) queremos listar todas las soluciones factibles.

- (a) Lo más importante es conseguir una cota pesimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.

- b) Lo más adecuado sería usar una técnica de ramificación y poda ya que es muy importante el orden en el que se exploran las soluciones parciales.
- c) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.

33. Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?

- a. se podría explorar menos nodos de los necesarios.
- b. Que se podría podar el nodo que conduce a la solución óptima.
- c. Que se podría explorar más nodos de los necesarios.

34. Un algoritmo recursivo basado en el esquema divide y vencerás...

- a. Las otras dos opciones son ambas verdaderas.
- b. ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a.
- c. ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.

35. Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones recurrente expresa mejor su complejidad temporal para el caso general, siendo n el número de discos?

- a) $T(n) = 2T(n - 1) + 1$
- b) $T(n) = 2T(n - 1) + n$
- c) $T(n) = T(n - 1) + n$

36. Una de las prácticas de laboratorio consistió en el cálculo empírico de la complejidad temporal promedio del algoritmo de ordenación de vectores Quicksort tomando como centinela el elemento del vector que ocupa la posición central. ¿Cuál es el orden de complejidad que se obtuvo?

- a. $n \log^2 n$
- b. n^2
- c. $n \log n$

Preg 37. Se desea ordenar una lista enlazada de n elementos haciendo uso del algoritmo Mergesort. En este caso, al tratarse de una lista, la complejidad temporal asintótica de realizar la división en subproblemas resulta ser lineal con el tamaño de esa lista. ¿Cuál sería entonces el coste temporal de realizar dicha ordenación?

- a) $\Theta(n \log n)$
- b) $\Theta(n^2)$

- c) Ninguna de las otras dos opciones es cierta.

Preg 38. Dada la siguiente función:

```
int exa (vector <int>& v) {  
    int i,sum=0, n=v.size();  
  
    if (n>0){  
        int j=n;  
        while (sum<100){  
            j=j/2;  
            sum=0;  
            for (i=j;i<n;i++)  
                sum+=v[i];  
            if (j==0) sum=100;  
        }  
        return j;  
    }  
    else return -1;  
}
```

Marcad la opción correcta.

- a) La complejidad temporal exacta es $\Theta(n \log n)$
- b) La complejidad temporal en el mejor de los casos es $\Omega(1)$
- c) La complejidad temporal en el mejor de los casos es $\Omega(n)$

39. Dado el problema del laberinto con tres movimiento ¿cuál de las estrategias siguientes proveería de una cota óptima para ramificación y poda?

- a. Suponer que ya no se van a realizar más movimientos
- b. **Las otras dos estrategias son ambas válidas**
- c. Suponer que en adelante todas las casillas del laberinto son accesibles

40. ¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?

- a. Que el algoritmo sería más lento pues se explotarian más nodos de los necesarios
- b. Nada especial, las cotas pesimistas no tienen por que corresponderse con soluciones factibles
- c. **Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.**

- 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
2. Tenemos un conjunto de n enteros positivos y queremos encontrar el subconjunto de tamaño m de suma mínima.
- a. Lo más adecuado sería usar una técnica de ramificación y poda, aunque en el peor caso el coste temporal asintótico (o complejidad temporal) sería exponencial.
 - b. Para encontrar la solución habría que probar con todas las combinaciones posibles de m enteros, con lo que la técnica de ramificación y poda no aporta nada con respecto a la vuelta atrás.
 - c. Una técnica voraz daría una solución óptima.
3. Di cuál de estos resultados de coste temporal asintótica es falsa:
- a. La ordenación de un vector usando el algoritmo QuickSort requiere en el peor caso un tiempo en $\Omega(n^2)$.
 - b. La ordenación de un vector usando el algoritmo MergeSort requiere en el peor caso un tiempo en $\Omega(n^2)$.
 - c. La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en $O(\log n)$.
4. La complejidad temporal (o coste temporal asintótico) en el mejor de los casos...
- a. ... es una función de la talla, o tamaño del problema, que tiene que estar definida para todos los posibles valores de ésta.
 - b. ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.
 - c. ... Las dos anteriores son verdaderas.
5. 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 pesas que no sean múltiplos de esta cantidad. Queremos hacer un programa que genere todas las combinaciones posibles. [A] o [C]
- a. No hay ningún problema en usar una técnica de vuelta atrás.
 - b. No se puede usar vuelta atrás porque las decisiones no son valores discretos.
 - c. No se puede usar vuelta atrás porque el número de combinaciones es infinitivo.
6. 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 basado en la mejor solución hasta el momento.

7. ¿Cuál de estas tres expresiones es cierta?

- a. $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$
- b. $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$
- c. $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$

8. Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2) + n$;

$f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:

- a. $f(n) \in \Theta(n^2)$
- b. $f(n) \in \Theta(n \log n)$
- c. $f(n) \in \Theta(n)$

9. Indicad cuál de estas tres expresiones es falsa:

- a. $\Theta(n/2) = \Theta(n)$
- b. $\Theta(n) \subset O(n)$
- c. $\Theta(n) \subset \Theta(n^2)$

10. Indica cuál es el coste temporal asintótico (o complejidad temporal), en función de n , del programa siguiente:

```
s=0; for(i=0; i<n; i++) for(j=i; j<n; j++) s += n*i+j;
```

- a. Es $O(n^2)$ pero no $\Omega(n^2)$.
- b. **Es $\Theta(n^2)$**
- c. Es $\Theta(n)$

11. Un programa con dos bucles anidados uno dentro del otro, cada uno de los cuales hace aproximadamente n iteraciones, tarda un tiempo

- a. **$O(n^2)$**
- b. $O(2^n)$
- c. $O(n)$

12. La eficiencia de los algoritmos voraces se basa en...

- a. ... el hecho de que, con antelación, las posibles decisiones se ordenan mejor a peor.
- b. ... el hecho de que las decisiones tomadas no se reconsideran
- c. ... En el esquema voraz no se puede hablar de eficiencia puesto que a menudo no resuelve el problema

13. Sea $f(n_i)$ la solución de la relación de recurrencia $f(n) = 2 f(n - 1) + 1$;

$f(1) = 1$. Indicad cuál de estas tres expresiones es cierta:

- a. $f(n) \in \Theta(n^2)$
- b. **$f(n) \in \Theta(2^n)$**

- c. $f(n) \in \Theta(n)$

14. Pertenece $3n^2 + 3$ a $O(n^3)$?

- a. No.
- b. Sólo para $c = 1$ y $n_0 = 5$
- c. Sí.

15. Las relaciones de recurrencia ...

- a. ... aparecen sólo cuando la solución sea del tipo divide y vencerás.
- b. ... expresan recursivamente el coste temporal de un algoritmo.
- c. ... sirven para reducir el coste temporal de una solución cuando es prohibitivo.

16. El coste temporal de un algoritmo se ajusta a la siguiente ecuación de recurrencia:

$$T(n) = \begin{cases} 1 & n = 0 \\ n + \sum_{j=0}^{n-1} T(j) & n > 1 \end{cases}$$

(pone $n = 0$)

¿qué coste temporal asintótico (o complejidad temporal) tendrán el algoritmo?

- a. $O(n \log(n))$
- b. $O(n^2)$
- c. $O(2^n)$

17. De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos: NO ES SEGURO

- a. El problema de cortar un tubo de forma que tenga que se obtenga el máximo beneficio posible
- b. El problema del cambio, o sea. el de encontrar la manera de entregar una cantidad de dinero usando las mínimas monedas.
- c. El problema del viajante de comercio

18. ¿Qué algoritmo es asintóticamente más rápido, el quickSort o el mergeSort?

- a. como su nombre indica, el quickSort
- b. son los dos iguales de rápidos, ya que el coste temporal asintótico de ambos es $O(n \log(n))$.
- c. el mergeSort es siempre más rápido o igual (salvo una constante) que el quickSort.

19. El coste temporal del algoritmo de ordenación por inserción es

- a. ... $O(n^2)$
- b. ... $O(n)$
- c. ... $O(n \log n)$

20. Los algoritmos de programación dinámica hacen uso...

- a. ... de que la solución óptima se puede construir añadiendo el componente óptimo de los restantes, uno a uno
- b. ... de que se puede ahorrar esfuerzo guardando los resultados de esfuerzos anteriores
- c. de una estrategia trivial consistente en examinar todas las soluciones

21. ¿Cuál de estos tres problemas de optimización no tiene una solución voraz (greedy) que sea óptima?

- a. El problema de la mochila continua o con fraccionamiento.
- b. **El problema de la mochila discreta.**
- c. El árbol de cobertura de coste mínimo de un grafo conexo

22. El problema de la función compuesta mínima consiste en encontrar, a partir de un conjunto de funciones dadas, la secuencia mínima de composición de éstas que permita transformar un número n en otro m. Se quiere resolver mediante ramificación y poda ¿Cuál sería la forma más adecuada de representar las posibles soluciones?

- a. Mediante un vector de booleanos.
- b. Mediante un vector de reales.
- c. Este problema no se puede resolver usando ramificación y poda si no se fija una cota superior al número total de aplicaciones de funciones.

23. ¿Cuál de estas tres expresiones es falsa?

- a. $2n^2 + 3n + 1 \in O(n^3)$
- b. $n + n \log(n) \in \Omega(n)$
- c. **$n + n \log(n) \in \Theta(n)$**

24. La solución recursiva ingenua a un determinado problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

- a. ... un algoritmo del estilo ed divide y vencerás
- b. **... un algoritmo de programación dinámica**
- c. ... un algoritmo voraz

25. Un algoritmo recursivo basado en el esquema divide y vencerás...

Seleccione una:

- a. ... será más eficiente cuanto más equitativa sea la división en subproblemas
- b. Las demás opciones son verdaderas.
- c. ... nunca tendrá una complejidad exponencial.

26. ¿Cual de estas expresiones es falsa?

Selecciones una

- a. $3n^2 + 1 \in O(n^3)$
- b. $n + n \log(n) \in \Omega(n)$
- c. $n + n \log(n) \in \Theta(n)$

27. Indica cuál es la complejidad en el peor caso de la función replace:

```
unsigned bound( const vector<int> &v ) {
    for( unsigned i = 0; i < v.size(); i++ )
        if( v[i] == '0' )
            return i;
    return v.size();
}

void replace( vector <int> &v ) {
    for( unsigned i = 0; i < bound(v); i++ )
        v[i] = c;
}
```

- a. $O(n \log n)$
- b. $\Theta(n^2)$
- c. $O(n)$

28. ¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n) {
    if (n<=1)
        return 0;
    unsigned sum = desperdicio (n/2) + desperdicio (n/2);
    for (unsigned i=1; i < n-1; i++)
        for (unsigned j=1; j<=i; j++)
            sum += i*j;
    return sum;
}

a.  $\Theta(n^2)$ 
b.  $\Theta(2^n)$ 
c.  $\Theta(n^2 \log n)$ 
```

29. Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2 f(n/2) + 1$; $f(1) = 1$.

Indicad cuál de estas tres expresiones es cierta:

- a. $f(n) \in \Theta(n)$
- b. $f(n) \in \Theta(n^2)$
- c. $f(n) \in \Theta(n \log(n))$

30. Considerad estos dos fragmentos:

```
s = 0; for (i = 0; i < n; i++) s+=i;
y
s = 0; for (i = 0; i < n; i++) if (s[i] != 0) s += i;
```

y un array $a[i]$ de números enteros.

Indicad cuál de estas tres afirmaciones es cierta:

Seleccione una:

- a. El coste temporal asintótico del primer programa en el caso peor es más alto que en el segundo.
- b. El coste temporal asintótico, tanto en el caso mejor como en el caso peor, de los dos programas es el mismo.
- c. El coste temporal asintótico del segundo programa en el caso peor es más alto que en el primero

31. Indica cuál es la complejidad, en función de n, del fragmento siguiente:

```
int a = 0;
for ( int i = 0; i < n; i++)
    for (int j = i; j > 0; j /= 2 )
        a += A[i] [j];
```

Seleccione una:

- a. $O(n \log n)$
- b. $O(n)$
- c. $O(n^2)$

32. Indica cuál es la complejidad en función de n, donde k es una constante (no depende de n), del fragmento siguiente:

```
for( int i = k; i < n - k; i++) {
    A[i] = 0;
    for( int j = i-k; j < i + k; j++)
        A[i] += B[j];
}
```

- a. $O(n)$
- b. $O(n \log n)$
- c. $O(n^2)$

33. La versión de QuickSort que utiliza como pivote la mediana del vector ...

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. ... se comporta peor cuando el vector ya está ordenado.
- c. ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo

34. Un programa con dos bucles anidados uno dentro del otro. El primero hace n iteraciones aproximadamente y el segundo la mitad, tarda un tiempo.

- a. $O(n \log n)$
- b. $O(n^2)$
- c. $O(n \sqrt{n})$

35. Un problema de tamaño n puede transformarse en tiempo $O(n^2)$ en nueve de tamaño $n/3$ por otro lado , la solución al problema cuando la talla es 1 requiere un tiempo constante.

¿Cuál de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a. $O(n \log n)$
- b. $O(n^2 \log n)$
- c. $O(n^2)$

36. ¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n) {
    if (n<=1)
        return 0;
    unsigned sum = desperdicio (n/2) + desperdicio (n/2);
    for (unsigned i=1; i < n-1; i++)
        for (unsigned j=1; j<=i; j++)
            for (unsigned k = 1; k<= j; k++)
                sum += i*j*k;
    return sum;
}
```

- a. $\Theta(2^n)$
- b. $\Theta(n^3 \log n)$
- c. $\Theta(n^3)$

37. Indica cuál es la complejidad de la función siguiente:

```
unsigned sum( const mat &A ) {
    unsigned d = A.n_rows();
    unsigned a = 0;
    for( unsigned i = 0; i < d; i++)
        for( unsigned j = 0; j < d; j++)
            a += A(i, j);
    return a;
}
```

- a. $O(n^2)$
- b. $O(n)$
- c. $O(n \log n)$

38. Los algoritmos de programación dinámica hacen uso...

- a. ... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.
- b. ... de que se puede ahorrar cálculos guardando resultados anteriores en un almacén
- c. ... de una estrategia trivial consistente en examinar todas las soluciones posibles

39. Cuando se calculan los coeficientes binomiales usando la recursión

$$\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$$

con

$$\binom{n}{0} = \binom{n}{n} = 1.$$

qué problema se da y cómo se puede resolver?

- a. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.
- b. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.
- c. **Se repiten muchos cálculos y ello se puede evitar usando programación dinámica**

40. Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/2)+n$, $f(1)=1$ indicad cuál de estas:

- a. $f(n) \in \Theta(n^2)$
- b. $f(n) \in \Theta(n)$
- c. **$f(n) \in \Theta(n \log(n))$**

41. Para que la complejidad de un algoritmo presente caso mejor y peor distintos...

- a. ... es condición necesaria y suficiente que existan instancias distintas del problema con el mismo tamaño.
- b. **... es condición necesaria que existan instancias distintas del problema con el mismo tamaño**
- c. ... es condición suficiente que existan instancias distintas del problema con el mismo tamaño

42. Un problema de tamaño n puede transformarse en tiempo $O(n)$ en siete de tamaño $n/7$ por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante ¿cuál de estas clases de coste temporal asintótico es la más ajustada?

- a. $O(n^2)$
- b. $O(n)$
- c. **$O(n \log n)$**

43. La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

- a. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo
- b. **Las demás opciones son falsas.**
- c. ... siempre coincidirá con la complejidad temporal de las instancias que están en el caso base del algoritmo recursiva.

Página 41 del otro pdf

Considerad la función siguiente:

```
int M(int i, int f){  
    if (i == f)  
        return i;  
    else{  
        e = v [ M(1, (i+f) /2)];  
        f = v [ M( (i + f)/ 2+1, f))]  
        if (e<f)  
            return e;  
        else  
            return f;  
    }  
}
```

Si la talla del problema viene dada por $n = f-i+1$ ¿cuál es el coste temporal asintótico en el supuesto de que n sea una potencia de 2?

Selecciona una:

- a. **$O(n)$**
- b. $O(n^2)$

c. $O(n \log(n))$

El coste temporal asintótico del fragmento

`s=0; for(i=0; i<n; i++) (for(j=i; j<n; j++) s+=i*j;`

y el del fragmento

`s=0; for(i=0;i<n;i++) for(j=0; j<n; j++);`

son ...

Seleccione una:

- a.iguales.
- b.el del segundo, menor el del primero.
- c.el del primero, menor que el del segundo.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n = 1 \\ n^2 + 3f(n/3) & n > 1 \end{cases}$$

Seleccione una:

- a. $f(n) \in \Theta(n^2)$
- b. $f(n) \in \Theta(n^2 \log n)$
- c. $f(n) \in \Theta(n)$

La versión de Quicksort que utiliza como pivote el elemento del vector que ocupa la primera posición...

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. ...se comporta peor cuando el vector ya está ordenado.
- c. ...El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

La versión de Quicksort que utiliza como pivote el elemento del vector que ocupa la posición central...

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. ... se comporta peor cuando el vector ya está ordenado.

- c. no presenta casos mejor y peor distintos para instancias del mismo tamaño.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

Dada la siguiente relación de recurrencia. ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n=1 \\ n+3f(n/3) & n>1 \end{cases}$$

Selecciona una:

- a. $f(n) \in \Theta(n \log n)$
- b. $f(n) \in \Theta(n^3)$
- c. $f(n) \in \Theta(n)$

Cuando se resuelve usando un algoritmo de vuelta atrás un problema de n decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?

Selecciona una.

- a. $O(2n^2)$
- b. $O(n!)$
- c. $O(n^2)$

Al resolver el problema del viajante de comercio mediante vuelta atrás y asumiendo un grafo de n vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

- a. Se multiplican por la distancia de la arista más corta que nos queda por considerar.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las n aristas más cortas.
- c. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido

La complejidad en el peor de los casos de un algoritmo de ramificación y poda...

Selecciona una:

- a) ... puede ser exponencial con el número de alternativas por cada decisión.
- b) ... puede ser polinómica con el número de decisiones a tomar.
- c) ... es exponencial con el número de decisiones a tomar.

La estrategia de ramificación y poda genera las soluciones posibles al problema mediante...

Seleccione una:

- a) ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.
- b) ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- c) ... un recorrido en anchura el árbol que representa el espacio de soluciones.

¿Para qué sirven las cotas pesimistas en ramificación y poda?

Seleccione una:

- a) Para tener la certeza de que la cota optimista está bien calculada.
- b) Para descartar nodos basándose en la preferencia por algún otro nodo ya completado.
- c) Para descartar nodos basándose en el beneficio esperado.

La ventaja de la estrategia ramificación y poda frente a vuelta atrás es que la primera genera las soluciones posibles al problema mediante ...

Seleccione una:

- a) ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.
- b) **Las otras dos opciones son verdaderas.**
- c) un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.

Tratándose de un problema de optimización, en la lista de nodos vivos de ramificación y poda ...

Seleccione una:

- a) ... sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento.
- b) ... puede haber nodos que no son prometedores.
- c) **Las otras dos opciones son ciertas.**

Cuando resolvemos un problema mediante un esquema de ramificación y poda

Seleccione una:

- a) ... los valores entre los cuales se elige en cada una de las decisiones tienen que formar un conjunto finito.
- b) ... las decisiones sólo pueden ser binarias.
- c) ... los valores entre los cuales se elige en cada una de las decisiones pueden formar un conjunto infinito.

La estrategia de ramificación y poda necesita cotas pesimistas...

Seleccione una:

- a) ... para decidir el orden de visita de los nodos del árbol de soluciones.
- b) sólo si se usa para resolver problemas de optimización.
- c) ... para determinar si una solución es factible.

En la estrategia de ramificación y poda ...

Seleccione una:

- a) ... cada nodo tiene su propia cota pesimista y también su propia cota optimista
- b) ... cada nodo tiene su propia cota pesimista, la cota optimista sin embargo, es común para todos los nodos.
- c) ... cada nodo tiene su propia cota optimista, la cota pesimista sin embargo, es común para todos los nodos.

Si para resolver un mismo problema usamos un algoritmo de ramificación y poda y lo modificamos mínimamente para convertirlo en un algoritmo de vuelta atrás, ¿qué cambiamos realmente?

Seleccione una:

- a) Cambiamos la función que damos a la cota pesimista.
- b) Provocamos que las cotas optimistas pierdan eficacia.
- c) Sería necesario comprobar si las soluciones son factibles o no puesto que ramificación y poda sólo genera nodos factibles.

En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás ...

Seleccione una:

- a) ... no se puede usar para resolver problemas de optimización.
- b) ... debe recorrer siempre todo el árbol.
- c) ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.

La estrategia de vuelta atrás es aplicable a problemas de selección y optimización en los que:

Seleccione una:

- a) El espacio de soluciones puede ser tanto finito como infinito pero en este último caso debe ser al menos numerable.
- b) El espacio de soluciones es un conjunto infinito.
- c) El espacio de soluciones es un conjunto finito.

Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila

Seleccione una:

- a) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- c) El valor óptimo de la mochila continua correspondiente.

Decid cuál de estas tres no sirve como cota optimista para obtener el valor óptimo de la mochila discreta:

Seleccione una:

- a) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- b) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- c) El valor de la mochila continua correspondiente.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ellos 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 vanas ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras que esté más cercana al destino en línea recta.

¿Que tipo de cota seria?

- a. Seria una cota pesimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.
- b. Ninguna de las otras dos opciones
- c. Seria una cota optimista siempre que se tenga la certeza ed que esa aproximación encuentra una solución factible

Se desea encontrar el camino mas corto entre dos ciudades.

Para ellos 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 vanas ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras que esté más cercana al destino según su distancia geográfica;

Este algoritmo voraz, ¿Serviría como cota pesimista?

- a. No, ya que no asegura que se encuentre una solución factible
- b. Sí, puesto que la distancia geográfica asegura que otra solución mejor no es posible
- c. No, ya que en algunos casos puede dar distancias menores que la óptima

El problema de cortar un tubo de longitud n en segmentos de longitud entera, de esta manera que el precio total de usu partes sea máximo de acuerdo con una lista de precios por longitudes....

- a. ... no se puede resolver mediante un algoritmo de vuelta atrás
- b. ... se puede resolver mediante un algoritmo de vuelta atrás pero existe una solución asintóticamente mucho más eficiente

- c. ... se debe resolver mediante un algoritmo de vuelta atrás, dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo

Di cuál de estas soluciones a problemas de optimización no comporte, en el peor caso, tener que considerar $O(n!)$ posibilidades.

Seleccione una

- a. La solución de vuelta atrás al problema del viajante de comercio (traveling salesman problem), o sea, el de encontrar un ciclo hamiltoniano de coste mínimo en un grafo conexos de n vértices donde cada artista 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 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
- c. La solución al problema de buscar un árbol que cubre todos los vértices de un grafo de n vértices de forma que el coste es mínimo (minimum spanning tree)

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos.

Selecciones una:

- a. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas
- b. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores
- c. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.

Un informático quiere subir a una montaña y para ellos decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además entenderán que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿que tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo voraz
- b. un algoritmo divide y vencerás
- c. un algoritmo de programación dinámica

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objeto?

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión de $O(n)$ a $O(1)$, donde n es el número de objetos a considerar
- b. Porque si no se hace es posible garantizar que la toma de decisiones siga un criterio voraz
- c. Para reducir la complejidad temporal en la toma de cada decisión: de $O(n^2)$ a $O(n \log n)$, donde n es el número temporal en la toma de cada decisión: de $O(n^2)$ a $O(n \log n)$, donde n es el número de objetos a considerar

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n=0 \\ \sum_{k=0}^{n-1} T(k) & n>0 \end{cases}$$

¿cuál de las siguiente afirmaciones es cierta?

Selecciones una:

- a. $T(n) \in \Theta(n^2)$
- b. $T(n) \in \Theta(n!)$
- c. $T(n) \in \Theta(2^n)$

En el método voraz ...

Seleccione una:

- a) ... siempre se encuentra solución pero puede que no sea la óptima.
- b) ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.
- c) ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

¿Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores?

Seleccione una:

- a) Ya no se garantizaría la solución óptima pero sí una factible.
- b) Habría que replantearse el criterio de una selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible.
- c) La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f( unsigned x, unsigned v[] ) {
    if (x==0)
        return 0;
    unsigned m = 0;
    for ( unsigned k = 0; k < x; k++ )
        m = max( m, v[k] + f( x-k, v ) );
    return m;
}
```

¿Cuál es la mejor estructura para el almacén? Seleccione una:

- a) int A
- b) int A[]
- c) int A[][]

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f( unsigned y, unsigned x){ // suponemos y >= x
    if (x==0 || y==x) return 1;
    return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor estructura para el almacén? Seleccione una:

- a) int A[]
- b) int A
- c) int A[][]

La solución de programación dinámica iterativa del problema de la mochila discreta

Seleccione una:

- a) calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.
- b) tiene un coste temporal asintótico exponencial con respecto al número de objetos
- c) tiene la restricción de que los valores tienen que ser enteros positivos.

El problema de encontrar el árbol de recubrimiento para un grafo no dirigido, conexo y ponderado...

- a) solo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo
- b) no se puede resolver en general con una estrategia voraz
- c) se puede resolver siempre con una estrategia voraz

Si ante un problema de decisión existe un criterio de selección voraz entonces...

Seleccione una:

- a) al menos una solución factible está garantizada
- b) Ninguna de las otras dos opciones es cierta
- c) La solución óptima está garantizada

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f( unsigned y, unsigned x){ // suponemos y >= x
    if (x==0 || y==x) return 1;
    return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a) $O(1)$
- b) $O(y^2)$
- c) $O(y)$

El valor que se obtiene con el método voraz para el problema de la mochila discreta es

- a) Una cota inferior para el valor óptimo que a veces puede ser igual a este
- b) Una cota inferior para el valor óptimo, pero que nunca coincide con este
- c) Una cota superior para el valor óptimo

¿Cuál de estas estrategias voraces obtiene un mejor valor para la mochila discreta?

- a) Meter primero los elementos de menor peso
- b) **Meter primero los elementos de mayor valor específico o valor por unidad de peso**
- c) Meter primero los elementos de mayor valor

Un tubo de n centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa. Seleccione una:

- a) Hacer una evaluación exhaustiva “de fuerza bruta” de todas las posibles maneras de cortar el tubo consume un tiempo $\Theta(2^n)$
- b) Es posible evitar hacer la evaluación exhaustiva “de fuerza bruta” guardando para cada posible longitud $j < n$ el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente
- c) **Hacer una evaluación exhaustiva “de fuerza bruta” de todas las posibles maneras de cortar el tubo consume un tiempo $\Theta(n!)$**

Se pretende implementar mediante programación dinámica iterativa la función recursiva

```
int f (int x, int y) {  
    if(x <= y) return 1;  
    return x + f(x-1, y);  
}
```

Cual es la mejor complejidad espacial que se puede conseguir?

- a) $O(y^2)$
- b) **$O(1)$**
- c) $O(y)$

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```

float f(unsigned x, int y) {
    if( y < 0 ) return 0;
    float A = 0.0;
    if ( v1[y] <= x )
        A = v2[y] + f( x-v1[y], y-1 );
    float B = f( x, y-1 );
    return min(A,2+B);
}

```

¿Cuál es la mejor complejidad espacial que se puede conseguir? Seleccione una:

- a) O(y^2)
- b) O(1)
- c) O(y)

La programación dinámica...

- a) En algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda en eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos
- b) **Las otras dos opciones son ciertas**
- c) normalmente se usa para resolver problemas de optimización con dominio discretizables puesto que las tablas se han de indexar con este tipo de valores

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución aportada por el método voraz?

- a) **El fontanero diligente y la mochila continua**
- b) El fontanero diligente y la asignación de tareas
- c) El fontanero diligente y el problema del cambio

Dado un problema de optimización, el método voraz...

- a) siempre se obtiene una solución factible
- b) siempre obtiene la solución óptima
- c) **garantiza la solución óptima sólo para determinados problemas**

Respuestas para la modalidad 1

1. 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) Divide y vencerás.
 - (b) Programación dinámica.
 - (c) El método voraz.
2. Dado un problema de optimización, el método voraz...
 - (a) ... siempre obtiene la solución óptima.
 - (b) ... siempre obtiene una solución factible.
 - (c) ... garantiza la solución óptima sólo con determinados problemas.
3. 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) $O(n)$.
 - (b) $O(\log n)$.
 - (c) $O(n \log n)$.
4. 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.
5. La complejidad temporal (o coste temporal asintótico) en el mejor de los casos...
 - (a) ... es una función de la talla, o tamaño del problema, que tiene que estar definida para todos los posibles valores de ésta.
 - (b) ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.
 - (c) Las otras dos opciones son ambas verdaderas.
6. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...
 - (a) ... pueden eliminar estados que conducen a soluciones factibles.
 - (b) ... garantizan que no se va a explorar todo el árbol de estados posibles.
 - (c) Las otras dos opciones son ambas verdaderas.

7. De las expresiones siguientes, 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 diferente de las otras dos.
- (a) Si $f \notin \Omega(g)$ entonces $O(f) = \Omega(g)$.
 - (b) Si $f \in O(g)$ entonces $g \notin O(f)$.
 - (c) Si $f \in \Theta(g)$ entonces $O(f) = O(g)$.
8. La eficiencia de los algoritmos voraces se basa en...
- (a) ... el hecho de que, con antelación, las posibles decisiones se ordenan de mejor a peor.
 - (b) ... el hecho de que las decisiones tomadas no se reconsideran.
 - (c) En el esquema voraz no se puede hablar de eficiencia puesto que a menudo no resuelve el problema.
9. 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.
10. Di cuál de estos tres problemas de optimización no comporta, en el peor caso, tener que considerar $O(n!)$ posibles soluciones.
- (a) El problema de la asignación de n tareas a n trabajadores de forma que cada trabajador hace exactamente una tarea y cada tarea es asignada a un trabajador exactamente, de forma que la suma de los costes de las tareas es mínimo.
 - (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 (*minimum spanning tree*).
11. Sea $T(n) = n + T(n - 1)$ para $n > 1$ y $T(1) = 1$. Una de las afirmaciones siguientes es cierta. Cuál?
- (a) $T(n) \in O(n^3)$
 - (b) $T(n) \in O(n \log n)$
 - (c) $T(n) \in O(n)$

12. Una de las siguientes afirmaciones es falsa. Cuál?

- (a) $O(2^n) = O(3^n)$
- (b) $O(n \log n) \subseteq O(n^2)$
- (c) $O(n^2 + 2n + 1) = O(n^2)$

13. Se pretende resolver el problema del viajante de comercio (*travelling salesman problem*) mediante Ramificación y poda. ¿Cuál de las siguientes acciones resulta ser mejor cota optimista para aplicarla a los nodos intermedios?

- (a) Obtener el árbol de recubrimiento de mínimo coste a los vértices aún no visitados.
- (b) Completar el recorrido pendiente mediante un algoritmo voraz.
- (c) Asumir que ya no quedan más vértices por visitar y cerrar el camino desde el último vértice visitado.

14. Se pretende implementar mediante programación dinámica iterativa la siguiente función recursiva:

```
int f(int m, int n, int p, int * v){  
    if( n <0 ) return 0;  
    int aux = 0;  
    if ( v[n] <= m )  
        aux = p + f( m-v[n], n-1, p, v );  
    return aux + f( m, n-1, p, v );  
}
```

¿Cuál sería la complejidad temporal del algoritmo iterativo?

- (a) $\Theta(m)$
- (b) $\Theta(m \cdot n)$
- (c) Ninguna de las otras dos opciones es la correcta.

15. Se pretende obtener la complejidad temporal en el caso más desfavorable de la siguiente función.

```
int exa (vector <int>& v) {
    int i,sum=0, n=v.size();

    if (n>0){
        int j=n;
        while (j>0 && sum<100){
            j=j/2;
            sum=0;
            for (i=j;i<n;i++)
                sum+=v[i];
        }
        return j;
    }
    else return -1;
}
```

¿Cuál de las siguientes formulaciones expresa mejor dicho coste?

(a) $c_s(n) = n \sum_{j=1}^{\log n} \sum_{i=1}^j \left(\frac{1}{2}\right)^i \in O(n \log n)$

(b) $c_s(n) = \sum_{j=0}^{n/2} \left(\frac{1}{2} \sum_{i=j}^n 1 \right) \in O(n \log n)$

(c) $c_s(n) = \sum_{j=0}^{n/2} \sum_{i=j}^n 1 \in O(n^2)$

16. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.

(a) Lo más adecuado sería usar una técnica de ramificación y poda ya que permite definir estrategias de exploración del árbol de estados posibles.

(b) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.

(c) Lo más importante es conseguir una cota optimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.

17. 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) 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).
 - (b) No: el cálculo de la fila k de la matriz (correspondiente a haber considerado k objetos) necesitamos todas las $k - 1$ filas anteriores.
 - (c) Sí, porque la solución es realmente un algoritmo voraz con complejidad espacial constante.
18. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?
- (a) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
 - (b) Ramificación y poda, puesto que con una estrategia apropiada es más eficiente que vuelta atrás.
 - (c) Vuelta atrás, es el esquema más eficiente para este problema.
19. Supongamos que estamos resolviendo un problema de optimización mediante un algoritmo de búsqueda y enumeración (por ejemplo, vuelta atrás o ramificación y poda). Las cotas optimistas se obtienen...
- (a) ... relajando, una o más de una, las restricciones que determinan si una solución es factible.
 - (b) ... relajando obligatoriamente *todas* las restricciones que determinan si una solución es factible.
 - (c) ... usando una solución subóptima (por ejemplo, voraz) para los elementos restantes de la solución.
20. Cuál es la complejidad temporal en el peor de los casos de construir un montículo (*heap*) a partir de un vector de tamaño n ?
- (a) $O(n)$.
 - (b) $O(\log n)$.
 - (c) $O(n \log n)$.
21. Se pretende resolver un problema de maximización utilizando ramificación y poda y para ello se dispone de tres posibles acotas pesimistas. Cuál tendríamos que utilizar para tratar de reducir la cantidad de nodos a explorar?
- (a) La que obtenga valores más pequeños.
 - (b) La que obtenga valores más elevados.
 - (c) La que más se aproxime a la solución voraz del nodo inicial.

22. 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 acabo 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.

23. 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.

24. 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) $O(n^2)$

25. Cuál de estos conceptos pertenece a la misma categoría que *divide y vencerás, vuelta atrás o ramificación y poda*?

- (a) *programación dinámica.*
- (b) *memoización.*
- (c) *cota optimista.*

26. Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n = 1 \\ n + 2f(n - 1) & n > 1 \end{cases}$$

- (a) $f(n) \in \Omega(2^n)$
- (b) $f(n) \in O(2^n)$
- (c) $f(n) \in \Theta(2^n)$

27. Una de las afirmaciones siguientes es falsa. Cuál?

- (a) Si $f(n) = 1 + f(\frac{n}{2})$ para $n > 1$ y $f(1) = 1$, entonces $f \in \Theta(n)$.
- (b) Si $f(n) = 1 + f(\frac{n}{2})$ para $n > 1$ y $f(1) = 1$, entonces $f \in \Theta(\log n)$.
- (c) Si $f(n) = n + 2f(\frac{n}{2})$ para $n > 1$ y $f(1) = 1$, entonces $f \in \Theta(n \log n)$.

28. 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) ...siempre se obtienen cuando se visitan las hojas del árbol de búsqueda.
- (b) ...se pueden obtener sin visitar necesariamente las hojas del árbol de búsqueda.
- (c) ...es posible que no encontramos la solución óptima.

29. 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) $O(n)$
- (b) $O(n \log n)$
- (c) $O(n^2)$

30. ¿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 temporal: En ramificación y poda es más probable que una instancia pertenezca al caso peor.
- (b) 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.
- (c) 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.

31. Cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:

- (a) El valor de la mochila continua correspondiente .
- (b) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- (c) El valor de una mochila que contiene todos los objetos a considerar aunque se pase del peso máximo permitido.

32. Cuál es la complejidad espacial del algoritmo *Quicksort*?

- (a) $O(n)$.
- (b) $O(n \log n)$.
- (c) $O(1)$.

33. 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 algoritmos de Kruskal va construyendo un bosque que va uniendo añadiendo vértices hasta obtener un único árbol de recubrimiento.
- (c) Los algoritmos de Prim y de Kruskal mantienen en todo momento un único árbol de recubrimiento, que crece añadiendo aristas o vértices.

34. Qué aporta la técnica de ramificación y poda frente a la de vuelta atrás?

- (a) Eficiencia. Los algoritmos de ramificación y poda son más eficientes que los de vuelta atrás.
- (b) 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.
- (c) La posibilidad de analizar diferentes estrategias para seleccionar el siguiente nodo a expandir.

35. Se pretende ordenar un vector cuyos n elementos están organizados formando un montículo (*Heap*). Sin tener en cuenta el tiempo empleado para este preproceso, ¿Con qué coste temporal asintótico se podría realizar la ordenación?

- (a) $O(n)$.
- (b) Ninguna de las otras dos opciones es la correcta.
- (c) $O(n \log n)$.

36. 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.
37. Una de las siguientes afirmaciones es falsa. Cuál?
- (a) Si $f \in O(g)$ y $g \in O(f)$, entonces $f = g$
(b) Si $f \in O(g)$ y $g \in O(f)$, entonces $O(f) = O(g)$
(c) Si $f \in O(g)$ y $g \in O(h)$, entonces $f \in O(h)$
38. 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 es estrictamente inferior a $O(n \log n)$.
(c) Porque el proceso de selección de los elementos que se incorporarán a la solución es siempre $O(n \log n)$.
39. 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) $O(n)$
(b) $O(n^2)$
(c) Ninguna de las otras dos opciones es la correcta.

40. La relación de recurrencia que se muestra expresa la complejidad de un algoritmo recursivo, donde $g(n)$ es una función polinómica:

$$T(n) = \begin{cases} 1 & n \leq 1 \\ T\left(\frac{n}{2}\right) + g(n) & n > 1 \end{cases}$$

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 diferente a las demás.

- (a) Si $g(n) \in O(n)$ 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 $g(n) \in O(1)$ la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.
- (c) Si $g(n) \in O(n^2)$ la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.

Respuestas para la modalidad 2

1. Se pretende implementar mediante programación dinámica iterativa la siguiente función recursiva:

```
int f(int m, int n, int p, int * v){  
    if( n <0 ) return 0;  
    int aux = 0;  
    if ( v[n] <= m )  
        aux = p + f( m-v[n], n-1, p, v );  
    return aux + f( m, n-1, p, v );  
}
```

¿Cuál sería la complejidad temporal del algoritmo iterativo?

- (a) $\Theta(m)$
 (b) $\Theta(m \cdot n)$
(c) Ninguna de las otras dos opciones es la correcta.

2. Cuál de estos conceptos pertenece a la misma categoría que *divide y vencerás, vuelta atrás o ramificación y poda?*

- (a) *programación dinámica.*
(b) *memoización.*
(c) *cota optimista.*

3. 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) $O(n)$
(b) $O(n^2)$
(c) Ninguna de la otras dos opciones es la correcta.

4. 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) $O(n)$.
 (b) $O(\log n)$.
(c) $O(n \log n)$.

5. Se pretende ordenar un vector cuyos n elementos están organizados formando un montículo (*Heap*). Sin tener en cuenta el tiempo empleado para este preproceso, ¿Con qué coste temporal asintótico se podría realizar la ordenación?

- (a) $O(n)$.
(b) Ninguna de las otras dos opciones es la correcta.
 (c) $O(n \log n)$.

6. Se pretende obtener la complejidad temporal en el caso más desfavorable de la siguiente función.

```

int exa (vector <int>& v) {
    int i,sum=0, n=v.size();

    if (n>0){
        int j=n;
        while (j>0 && sum<100){
            j=j/2;
            sum=0;
            for (i=j;i<n;i++)
                sum+=v[i];
        }
        return j;
    }
    else return -1;
}

```

¿Cuál de las siguientes formulaciones expresa mejor dicho coste?

(a) $c_s(n) = n \sum_{j=1}^{\log n} \sum_{i=1}^j \left(\frac{1}{2}\right)^i \in O(n \log n)$

(b) $c_s(n) = \sum_{j=0}^{n/2} \left(\frac{1}{2} \sum_{i=j}^n 1 \right) \in O(n \log n)$

(c) $c_s(n) = \sum_{j=0}^{n/2} \sum_{i=j}^n 1 \in O(n^2)$

7. Cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:

(a) El valor de la mochila continua correspondiente .

(b) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

(c) El valor de una mochila que contiene todos los objetos a considerar aunque se pase del peso máximo permitido.

8. Una de las afirmaciones siguientes es falsa. Cuál?

(a) Si $f(n) = 1 + f(\frac{n}{2})$ para $n > 1$ y $f(1) = 1$, entonces $f \in \Theta(n)$.

(b) Si $f(n) = 1 + f(\frac{n}{2})$ para $n > 1$ y $f(1) = 1$, entonces $f \in \Theta(\log n)$.

(c) Si $f(n) = n + 2f(\frac{n}{2})$ para $n > 1$ y $f(1) = 1$, entonces $f \in \Theta(n \log n)$.

9. 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.

10. 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) 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).
- (b) No: el cálculo de la fila k de la matriz (correspondiente a haber considerado k objetos) necesitamos todas las $k - 1$ filas anteriores.
- (c) Sí, porque la solución es realmente un algoritmo voraz con complejidad espacial constante.
11. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
- (a) Lo más adecuado sería usar una técnica de ramificación y poda ya que permite definir estrategias de exploración del árbol de estados posibles.
- (b) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
- (c) Lo más importante es conseguir una cota optimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.
12. 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) ...siempre se obtienen cuando se visitan las hojas del árbol de búsqueda.
- (b) ...se pueden obtener sin visitar necesariamente las hojas del árbol de búsqueda.
- (c) ...es posible que no encontramos la solución óptima.
13. Supongamos que estamos resolviendo un problema de optimización mediante un algoritmo de búsqueda y enumeración (por ejemplo, vuelta atrás o ramificación y poda). Las cotas optimistas se obtienen...
- (a) ...relajando, una o más de una, las restricciones que determinan si una solución es factible.
- (b) ...relajando obligatoriamente *todas* las restricciones que determinan si una solución es factible.
- (c) ...usando una solución subóptima (por ejemplo, voraz) para los elementos restantes de la solución.

14. La complejidad temporal (o coste temporal asintótico) en el mejor de los casos...
- (a) ... es una función de la talla, o tamaño del problema, que tiene que estar definida para todos los posibles valores de ésta.
 - (b) ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.
 - (c) Las otras dos opciones son ambas verdaderas.
15. 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.
16. 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 acabo 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.
17. Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n = 1 \\ n + 2f(n - 1) & n > 1 \end{cases}$$

- (a) $f(n) \in \Omega(2^n)$
- (b) $f(n) \in O(2^n)$
- (c) $f(n) \in \Theta(2^n)$

18. Se pretende resolver el problema del viajante de comercio (*travelling salesman problem*) mediante Ramificación y poda. ¿Cuál de las siguientes acciones resulta ser mejor cota optimista para aplicarla a los nodos intermedios?

- (a) Obtener el árbol de recubrimiento de mínimo coste a los vértices aún no visitados.
- (b) Completar el recorrido pendiente mediante un algoritmo voraz.
- (c) Asumir que ya no quedan más vértices por visitar y cerrar el camino desde el último vértice visitado.

19. 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) $O(n^2)$

20. 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 es estrictamente inferior a $O(n \log n)$.
- (c) Porque el proceso de selección de los elementos que se incorporarán a la solución es siempre $O(n \log n)$.

21. 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.

22. 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) $O(n)$
- (b) $O(n \log n)$
- (c) $O(n^2)$

23. Di cuál de estos tres problemas de optimización no comporta, en el peor caso, tener que considerar $O(n!)$ posibles soluciones.

- (a) El problema de la asignación de n tareas a n trabajadores de forma que cada trabajador hace exactamente una tarea y cada tarea es asignada a un trabajador exactamente, de forma que la suma de los costes de las tareas es mínimo.
- (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 (*minimum spanning tree*).

24. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

- (a) ... pueden eliminar estados que conducen a soluciones factibles.
- (b) ... garantizan que no se va a explorar todo el árbol de estados posibles.
- (c) Las otras dos opciones son ambas verdaderas.

25. Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Qué esquema es el más adecuado?

- (a) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
- (b) Ramificación y poda, puesto que con una estrategia apropiada es más eficiente que vuelta atrás.
- (c) Vuelta atrás, es el esquema más eficiente para este problema.

26. Se pretende resolver un problema de maximización utilizando ramificación y poda y para ello se dispone de tres posibles acotas pesimistas. Cuál tendríamos que utilizar para tratar de reducir la cantidad de nodos a explorar?

- (a) La que obtenga valores más pequeños.
- (b) La que obtenga valores más elevados.
- (c) La que más se aproxime a la solución voraz del nodo inicial.

27. La relación de recurrencia que se muestra expresa la complejidad de un algoritmo recursivo, donde $g(n)$ es una función polinómica:

$$T(n) = \begin{cases} 1 & n \leq 1 \\ T\left(\frac{n}{2}\right) + g(n) & n > 1 \end{cases}$$

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 diferente a las demás.

- (a) Si $g(n) \in O(n)$ 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 $g(n) \in O(1)$ la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.
- (c) Si $g(n) \in O(n^2)$ la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.

28. Una de las siguientes afirmaciones es falsa. Cuál?

- (a) $O(2^n) = O(3^n)$
- (b) $O(n \log n) \subseteq O(n^2)$
- (c) $O(n^2 + 2n + 1) = O(n^2)$

29. La eficiencia de los algoritmos voraces se basa en...

- (a) ... el hecho de que, con antelación, las posibles decisiones se ordenan de mejor a peor.
- (b) ... el hecho de que las decisiones tomadas no se reconsideran.
- (c) En el esquema voraz no se puede hablar de eficiencia puesto que a menudo no resuelve el problema.

30. Cuál es la complejidad temporal en el peor de los casos de construir un montículo (*heap*) a partir de un vector de tamaño n ?

- (a) $O(n)$.
- (b) $O(\log n)$.
- (c) $O(n \log n)$.

31. Cuál es la complejidad espacial del algoritmo *Quicksort*?
- (a) $O(n)$.
 - (b) $O(n \log n)$.
 - (c) $O(1)$.
32. 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.
33. ¿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 temporal: En ramificación y poda es más probable que una instancia pertenezca al caso peor.
 - (b) 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.
 - (c) 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.
34. De las expresiones siguientes, 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 diferente de las otras dos.
- (a) Si $f \notin \Omega(g)$ entonces $O(f) = \Omega(g)$.
 - (b) Si $f \in O(g)$ entonces $g \notin O(f)$.
 - (c) Si $f \in \Theta(g)$ entonces $O(f) = O(g)$.
35. Sea $T(n) = n + T(n - 1)$ para $n > 1$ y $T(1) = 1$. Una de las afirmaciones siguientes es cierta. Cuál?
- (a) $T(n) \in O(n^3)$
 - (b) $T(n) \in O(n \log n)$
 - (c) $T(n) \in O(n)$

36. 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 algoritmos de Kruskal va construyendo un bosque que va uniendo añadiendo vértices hasta obtener un único árbol de recubrimiento.
- (c) Los algoritmos de Prim y de Kruskal mantienen en todo momento un único árbol de recubrimiento, que crece añadiendo aristas o vértices.

37. Una de las siguientes afirmaciones es falsa. Cuál?

- (a) Si $f \in O(g)$ y $g \in O(f)$, entonces $f = g$
- (b) Si $f \in O(g)$ y $g \in O(f)$, entonces $O(f) = O(g)$
- (c) Si $f \in O(g)$ y $g \in O(h)$, entonces $f \in O(h)$

38. Dado un problema de optimización, el método voraz...

- (a) ... siempre obtiene la solución óptima.
- (b) ... siempre obtiene una solución factible.
- (c) ... garantiza la solución óptima sólo con determinados problemas.

39. 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) Divide y vencerás.
- (b) Programación dinámica.
- (c) El método voraz.

40. Qué aporta la técnica de ramificación y poda frente a la de vuelta atrás?

- (a) Eficiencia. Los algoritmos de ramificación y poda son más eficientes que los de vuelta atrás.
- (b) 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.
- (c) La posibilidad de analizar diferentes estrategias para seleccionar el siguiente nodo a expandir.

Queremos resolver mediante vuelta atrás el problema de las 8 reinas. Una buena cota optimista permitiría:

- = No es aplicable ese tipo de podas a este problema.
- ~ Muy probablemente, explorar menos nodos.
- ~ Muy probablemente, resolver el problema de forma mas rápida.

El algoritmo de ordenacion Quicksort divide el problema en dos subproblemas.¿Cuál es la complejidad temporal asintotica de realizar esa division?

- = $\Theta(n)$
- ~ $\Theta(n \log n)$
- ~ $\Theta(\log n)$

¿Qué complejidad se obtiene a partir de la relación de recurrencia $T(n)=9T(n/3) + n^3$ con $T(1) = O(1)$?

- = $O(n^3)$
- ~ $O(n^3 \log n)$
- ~ $O(n \log n)$

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo,

donde $g(n)$ es una función polinómica:

**$T(n) = 1$ si $n \leq 1$ ///
 $2T(n/2) + g(n)$ en otro caso**

Di cuál de las siguientes afirmaciones es cierta:

- ~ Si $g(n) \in \Theta(1)$ la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

= Si $g(n) \in \Theta(n^2)$ la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación mediante inserción binaria.

- ~ Si $g(n) \in \Theta(n)$ la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación mergesort.

Se pretende resolver el problema del viajante de comercio mediante el esquema de vuelta atrás, ¿Cuál de los siguientes valores se espera que se comporte mejor para decidir si un nodo es prometedor?

- = La suma de los pesos de las k aristas restantes mas cortas, donde k es el numero de ciudades que quedan por visitar.
- ~ El valor que se obtiene de multiplicar k por el peso de la arista mas corta de entre las restantes, donde k es el número de ciudades que quedan por visitar.
- ~ La suma de los pesos de las aristas que completan la solución paso a paso visitando el vértice mas cercano al ultimo visitado.

Si $f \notin O(g_1)$ y $f \in O(g_2)$ entonces siempre se cumplirá:

- = $f \notin O(\max(g_1, g_2))$
- ~ $f \in \Omega(g_1 + g_2)$
- ~ $f \in \Omega(\min(g_1, g_2))$

¿Cuál de las siguientes estrategias de búsqueda es mas apropiada en un esquema de vuelta atrás?

- = Ninguna de las otras dos estrategias es compatible con el esquema de vuelta atrás.
- ~ Explorar primero los nodos con mejor cota optimista.
- ~ Explorar primero los nodos con mejor valor hasta el momento en la función que se pretende optimizar.

¿que nos proporciona la media aritmética entre el coste temporal asintótico(o complejidad temporal) en el peor caso y el coste temporal asintótico en el mejor caso?

- ~ el coste temporal promedio
- ~ el coste temporal asintótico en el caso medio
- = nada de interés

Dada la siguiente función:

```
int exa (vector <int>& v) {  
    int j, i=i, 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.

- ~ La complejidad temporal en el mejor de los casos es $\Omega(1)$
- ~ La complejidad temporal exacta es $\Theta(n^2)$
- = La complejidad temporal en el mejor de los casos es $\Omega(n)$

Dada la siguiente función (donde $\max(a,b) \in \Theta(1)$):

```
float exa (vector<float>&v, vector<int>&p, int P, int i)
{
    float a, b;
    if (i>=0){
        if(p[i] <= P)
            a= v[i] + exa (v,p,P-p[i], i-1);
        else a = 0;
        b= exa(v,p,P,i-1);
    }
    return max(a,b);
}
```

Return 0;

Marcad la opción correcta.

- ~ La complejidad temporal en el mejor de los casos es $\Omega(n^2)$
- ~ La complejidad temporal en el peor de los casos es $O(n^2)$
- = La complejidad temporal en el peor de los casos es $O(2^n)$

Sea la siguiente relación de recurrencia:

$T(n) = 1 \text{ si } n \leq 1 // 8T(n/8) + g(n) \text{ en otro caso}$

Si $T(n) \in \Theta(n^2)$, ¿en cual de estos tres casos nos podemos encontrar?

~ $g(n) = n^3$

~ $g(n) = n$

= $g(n) = n^2$

¿Qué tienen en común los algoritmos de ordenación Quicksort y Mergesort?

~ La complejidad temporal de la combinación de las soluciones parciales.

~ La complejidad temporal de la división en subproblemas.

= El número de llamadas recursivas que hacen en el mejor de los casos.

Con respecto a la complejidad espacial de los algoritmos de ordenación Quicksort, Heapsort y Mergesort...

- ~ Mergesort y Heapsort tienen complejidad espacial lineal con el tamaño del vector a ordenar, la de Quicksort es constante.
- ~ La complejidad espacial de todos ellos es lineal con el tamaño del vector a ordenar.
- = Mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante.

Si $\lim_{n \rightarrow \infty} f(n)/g(n) = \infty$ entonces...

- ~ ... $f(n) \in O(g(n))$
- ~ ... $f(n) \in \Theta(g(n))$
- = ... $f(n) \in \Omega(g(n))$

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.

- ~ $\Omega(n^2) \subset \Omega(n)$
- ~ $n + n \log_2 n \in \Omega(n + n \log_2 n)$
- = $O(n^2) \subset O(2^{\log_2 n})$

Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es par y que se utilizará la siguiente recurrencia: $\text{pot}(x, n) = \text{pot}(x, n/2) * \text{pot}(x, n/2)$; ¿Qué esquema resultará ser más eficiente en cuanto al coste temporal?

- ~ En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.
- ~ Las otras dos opciones son ambas verdaderas
- = Un algoritmo recursivo con memoización

Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿Qué ocurre si la cota optimista resulta ser un valor excesivamente pequeño?

- ~ Que se podría explorar menos nodos de los necesarios.
- ~ Que se podría explorar el nodo que conduce a la solución óptima.
- = Que se podría explorar más nodos de los necesarios.

Decid cuál de estas tres estrategias proveería la cota pesimista más ajustada al valor óptimo de la mochila discreta:

- ~ El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
 - ~ Asumir que ya no se van a coger más objetos.
- = Completar las decisiones restantes basándose en la mejor solución voraz que pueda encontrarse para los restantes objetos y espacio disponible de mochila.

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.

- ~ $\Theta(n^2)$ C $\Theta(n^3)$
- = $O(n^2)$ C $O(n^3)$
- ~ $\Omega(n^2)$ C $\Omega(n^3)$

Dado el problema del laberinto con tres movimientos, ¿se puede aplicar un esquema de programación dinámica para obtener un camino de salida?

- ~ No, para garantizar que se encuentra un camino de salida hay que aplicar métodos de búsqueda exhaustiva como vuelta atrás o ramificación y poda.
- = Si, con este esquema siempre se puede encontrar un camino de salida si existe.
- ~ 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.

En un algoritmo de optimización resuelto mediante ramificación y poda ¿Podría encontrarse la solución óptima sin haber alcanzado nunca un nodo hoja?

- ~ Si, esto puede ocurrir incluso si no se hace uso de cotas pesimistas.
- = Si, pero esto solo podría ocurrir si se hace uso de cotas pesimistas.
- ~ No, los nodos hojas son los nodos completados y por lo tanto hay que visitar al menos uno de ellos para almacenarlo como la mejor solución hasta el momento.

Un problema de decisión en el que no hay función objetivo... (marca la opción correcta)

- ~ ... no puede ser resuelto mediante la técnica de ramificación y poda.
- = ... puede que tenga solución eficiente mediante programación dinámica.
- ~ ... solo puede ser resuelto de manera eficiente mediante un algoritmo voraz.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f (int x, int y){  
    If(x <= y)  
        Return 1;  
    Return x + f(x-1,y);  
}
```

¿Cuál son las mejores complejidades temporal y espacial que se pueden conseguir?

- ~ Temporal $O(x^2)$ y espacial $O(x)$
- = Temporal $O(x)$ y espacial $O(1)$
- ~ $O(x)$, tanto temporal como espacial

¿Cuál es la complejidad temporal de la siguiente función?

```
inf f (int n){  
    int k = 0;  
    for (int i = 1; i < n; i*=2)  
        for (int j = i; j > 0; j-=2)  
            k++;  
    return k;  
}  
~  $\Theta(n^2)$   
=  $\Theta(n)$   
~  $\Theta(n \log n)$ 
```

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f (unsigned int n){  
    if (n == 0)  
        return 1;  
    return f(n/2) + f(n/2) + n;  
}
```

¿Qué complejidades temporal y espacial cabe esperar de la función resultante?

~ Complejidad temporal $O(\log n)$ y espacial $O(n)$

= Complejidad temporal $O(\log n)$ y espacial $O(1)$

~ $O(n)$, tanto temporal como espacial

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 cierta.

~ Se puede construir una solución al problema basada en el esquema de ramificación y poda, pero el uso de cotas optimistas y pesimistas no mejoraría el algoritmo resultante.

~ La complejidad temporal de la mejor solución posible al problema es $O(n^2)$.

= La complejidad temporal de la mejor solución posible al problema es $O(n!)$

Una de estas tres afirmaciones es falsa. ¿Cuál?

~ Un algoritmo voraz puede no encontrar la solución óptima de un problema de selección discreta.

~ El esquema de ramificación y poda no garantiza que la complejidad temporal de resolución de un problema de selección discreta no sea exponencial.

= Los algoritmos voraces no sirven para resolver problemas de selección discreta.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$F(n) = 1 \text{ si } n=1 /// \sqrt{n} + 3f(n/3) \text{ si } n>1$

~ $f(n) \in \Theta(\sqrt{n} \log n)$

= $f(n) \in \Theta(n)$

~ $f(n) \in \Theta(n^3)$

De las siguientes tres afirmaciones, una es cierta y dos falsas, o bien una es falsa y dos son ciertas. Marca la que en ese sentido es diferente a las otras dos.

- ~ Ramificación y poda siempre es mas eficiente que vuelta atrás.
- ~ Ramificación y poda sirve para resolver problemas que vuelta atrás no puede.
- = Ramificación y poda resuelve el mismo tipo de problemas que vuelta atrás.

¿Cuál de los siguientes algoritmos de ordenación necesita un espacio de almacenamiento adicional al vector que se ordena con complejidad $O(n)$?

- ~ Quicksort
- ~ Bubblesort
- = Mergesort

Se dispone de un vector v que almacena números enteros en orden estrictamente creciente, y se desea averiguar si existe algún elemento que cumpla $v[i]=i$. De entre las estrategias que se citan, ¿Cuál sería la eficiente para resolver el problema?

- ~ Algoritmo voraz
- ~ Programación dinámica
- = Divide y vencerás

Una empresa de mensajería tiene n repartidores con distintos tiempos de entrega según el tipo de envío. Se trata de asignar los próximos n envíos, uno a cada repartidor, minimizando el tiempo total de todos los envíos. Para ello se conoce de antemano una tabla de tiempos en la que el valor t_{ij} corresponde al tiempo que emplea el repartidor i en realizar el envío j . De entre las estrategias que se citan, ¿Cuál sería la más eficiente para resolver el problema?

- ~ Vuelta atrás
- ~ Ramificación y poda
- = Algoritmo voraz

Indica cual de las siguientes afirmaciones es cierta:

- ~ Si un esquema de ramificación y poda encuentra la solución óptima a un problema, un esquema de vuelta atrás podría no encontrarla.
- ~ Si un esquema de ramificación y poda encuentra la solución óptima a un problema, un esquema de vuelta atrás siempre es más ineficiente.
- = Si un esquema de ramificación y poda encuentra la solución optima a un problema, un esquema de vuelta atrás también la encuentra siempre.

Dado el siguiente programa recursivo:

```
int f (int n) {    //Se asume que n>=0  
    if(n == 0)  
        return 1;  
    return f (n - 1) + f (n - 2);  
}
```

Si quisiéramos mejorarlo haciendo uso de la técnica de programación dinámica, ¿Cuáles serían las complejidades temporal y espacial más ajustadas del algoritmo resultante?

- ~ Ambas complejidades serían $O(1)$
- ~ Ambas complejidades serían $O(n)$
- = Respectivamente, $O(n)$ y $O(1)$

¿En qué caso la complejidad temporal del algoritmo de ordenación Quicksort es igual a la complejidad temporal del algoritmo mergesort?

- ~ Tanto en el caso peor como en el caso mejor de ambos
- ~ En el caso peor de ambos
- = En el caso mejor de ambos

Indica cual de las siguientes afirmaciones sobre el problema del fontanero diligente es cierta:

- ~ El esquema voraz podría no encontrar ninguna solución
- ~ Se puede demostrar que un esquema voraz siempre encuentra alguna solución
- = Se puede demostrar que un esquema voraz encuentra siempre la solución óptima

Sea el problema de la función compuesta mínima. Si no acotamos el número máximo de operaciones posibles, un esquema de ramificación y poda:

- ~ Si incluimos memorización, siempre encuentra la solución óptima.
- ~ Si incluimos memorización, podría no encontrar la solución óptima pero siempre acabaría.
- = Podría no acabar, al tener que expandir indefinidamente nuevos nodos.

Indica cuál de las siguientes afirmaciones es cierta:

- ~ Si un esquema de vuelta atrás encuentra la solución óptima a un problema, un esquema voraz también la encuentra siempre.
- ~ Si un esquema de vuelta atrás encuentra la solución optima a un problema, un esquema voraz no se puede plantear.
- = Si un esquema de vuelta atrás encuentra la solución optima a un problema, un esquema voraz también podría encontrarla.

¿En qué caso la complejidad temporal de Quicksort es la misma que la del algoritmo de ordenación por inserción?

- ~ En ningún caso
- ~ En el caso mejor
- = En el caso peor

Se quiere desarrollar un programa que compruebe si es posible que un caballo de ajedrez, mediante una secuencia de sus movimientos permitidos, recorra todas las casillas de un tablero N x N a partir de una determinada casilla dada como entrada y sin repetir ninguna casilla. De entre las estrategias que se citan, ¿Cuál sería la eficiente para resolver el problema?

- ~ Programación dinámica
- ~ Algoritmo voraz
- = Vuelta atrás

Indica cual de las siguientes afirmaciones sobre el problema de la mochila continua es cierta:

- ~ Un esquema voraz siempre encuentra alguna solución, aunque no sea optima.
- ~ El esquema voraz podría no encontrar ninguna solución.
- = Se puede demostrar que un esquema voraz encuentra siempre la solución óptima.

Si $\lim_{n \rightarrow \infty} (f(n)/n^2) = 0$ ¿cuál de estas tres afirmaciones es falsa?

- = $f(n) \in \Theta(n)$
- ~ $f(n) \in \Theta(n^3)$
- ~ $f(n) \in \Theta(n^2)$

En una carrera de coches por el desierto uno de los principales problemas es el abastecimiento de gasolina. Un participante dispone de un mapa que le indica las distancias entre gasolineras que hay en la ruta y cree que, parándose a repostar el menor numero de veces posible, podrá ganar. Para ayudarle hay que diseñar un algoritmo que le sugiera en que gasolineras debe hacerlo. Hay que tener en cuenta que hay una única ruta posible. De entre las estrategias que se citan, ¿Cuál sería la eficiente para resolver el problema?

= Programación dinámica

~ Algoritmo voraz

~ Ramificación y poda

La relación de recurrencia $T(n) = 1 + T(n - 1)$ si $n > 1$; $T(1) = 1$...

= Expresa el numero de llamadas recursivas que hace el algoritmo Quicksort en el peor de los casos.

~ Ninguna de las otras dos opciones es cierta

~ Expresa la complejidad temporal asintótica en el peor de los casos del algoritmo de búsqueda binaria.

Solo una de estas tres relaciones de recurrencia es tal que $T(n) \in \Theta(n)$. ¿Cuál?

= $T(n) = 1 + 2T(n / 2)$ si $n > 1$; $T(1) = 1$

~ $T(n) = n + T(n - 1)$ si $n > 1$; $T(1) = 1$

~ $T(n) = 1 + T(n / 2)$ si $n > 1$; $T(1) = 1$

De las siguientes tres afirmaciones, una es cierta y dos falsas, o bien una es falsa y dos son ciertas. Marca la que en ese sentido es diferente a las otras dos.

= Para que un problema tenga solución mediante programación dinámica es condición necesaria que pueda resolverse mediante divide y vencerás.

~ Todo problema que tiene solución mediante ramificación y poda también la tendrá mediante programación dinámica.

~ Todo problema que tiene solución mediante divide y vencerás también la tendrá mediante programación dinámica.

Sea el vector $v[8] = \{8,6,4,5,3,2,2\}$. Indica cual de las siguientes opciones es cierta. (Se asume C/C++ en la que el primer elemento del vector esta en la posición 0, es decir $v[0]$)

- = El vector v es un montículo máximo
- ~ El vector v no es un montículo máximo por que el elemento $v[3]=5$ debe ser “flotado” (desplazado hacia la izquierda)
- ~ El vector v no es un montículo máximo por que el elemento $v[2]=4$ debe ser “hundido” (desplazado hacia la derecha)

Dado un vector de tamaño n de números enteros. ¿Con que complejidad temporal se puede determinar si sus elementos están dispuestos formando un montículo de mínimos?

- = $O(n)$
- ~ $O(1)$
- ~ $O(\log n)$

Que complejidad temporal asintótica cabe esperar de un algoritmo de divide y vencerás cuya función descomponer produce, en tiempo constante, dos subproblemas iguales de tamaño $n-2$ cada uno y cuya función combinar es lineal con n , donde n es el tamaño del problema.

- = $O(2^n)$
- ~ $O(n \log n)$
- ~ $O(n^2)$

De las siguientes tres afirmaciones, una es cierta y dos falsas, o bien una es falsa y dos son ciertas. Marca la que en ese sentido es diferente a las otras dos.

- = El problema de la mochila discreta no se puede resolver mediante la técnica de divide y vencerás.
- ~ El problema de la mochila continua no se puede resolver mediante la técnica de ramificación y poda pues se podría producir un numero infinito de visitas.
- ~ El problema de la mochila continua no se puede resolver mediante la técnica de divide y vencerás pues se podría producir un numero infinito de llamadas recursivas.

Cuando un algoritmo recursivo que sigue el esquema divide y vencerás incurre en complejidades temporales prohibitivas por que se resuelven repetidamente los mismos subproblemas...

= ... podemos guardar soluciones parciales en un almacén para enviar esa repetición y puede ser que resolvamos menos problemas que si lo convertimos en iterativo.

~ ... podemos guardar soluciones parciales en un almacén para evitar esa repetición, pero resolveremos indefectiblemente mas problemas que si lo convertimos en iterativo.

~ ... debemos convertirlo obligatoriamente en iterativo para evitarlo.

Haciendo uso de la función merge del algoritmo mergesort se quiere mezclar k vectores ordenados de n elementos cada uno y obtener un único vector de kn elementos. Para ello primero se mezclan los dos primeros vectores, luego el resultado se mezcla con el tercero y a su vez, este resultado se mezcla con el cuarto y así hasta llegar al k-esimo. ¿Cuál sería la complejidad del algoritmo?

= $\Theta(n * k^2)$

~ $\Theta(n * k)$

~ $\Theta(n^2 * k)$

¿Cuál es la definición correcta de O(f)?

= $O(f) = \{g : N \rightarrow R + | \exists c \in R, \forall n_0 \in N, \forall n \geq n_0, f(n) \leq cf(n)\}$

~ $O(f) = \{g : N \rightarrow R + | \forall c \in R, \forall n_0 \in N, \forall n \geq n_0, g(n) \leq cf(n)\}$

~ $O(f) = \{g : N \rightarrow R + | \exists c \in R, \forall n_0 \in N, \forall n \geq n_0, f(n) \leq cg(n)\}$

Supongamos el problema de la mochila discreta modificado de manera que cada objeto puede seleccionarse, no seleccionarse o seleccionarse solo la mitad obteniendo en este caso la mitad de su valor. ¿Qué algoritmo resulta ser más eficiente para encontrar la solución óptima?

= Un algoritmo de ramificación y poda

~ Un algoritmo de vuelta atrás

~ Un algoritmo voraz

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$F(n) = 1 \text{ si } n = 1 // / n + 3f(n/3) \text{ si } n > 1$

= $f(n) = \Theta(n \log n)$

$\sim f(n) = \Theta(n^3)$

$\sim f(n) = \Theta(n)$

Sean las funciones $f_1(n) = n^{0.99999} \log n$, $f_2(n) = 1000000n$, $f_3(n) = 1.000001^n$, $f_4(n) = n^2$

Desde el punto de vista asintótico, ¿Cuál de las siguientes ordenaciones de mejor a peor es la correcta?

= $f_1(n); f_2(n); f_4(n); f_3(n)$

$\sim f_1(n); f_2(n); f_3(n); f_4(n)$

$\sim f_2(n); f_1(n); f_4(n); f_3(n)$

Sea $f(n)$ la solución de la relación de recurrencia $f(n) = 2f(n/1) + 1$; $f(1) = 1$. Indicad cuál

de estas tres expresiones es cierta:

= $f(n) \in \Theta(n)$

$\sim f(n) \in \Theta(n \log n)$

$\sim f(n) \in \Theta(n^2)$

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.

$\sim O(n^2) \subset O(n^3)$

= $\Theta(n^2) \subset \Theta(n^3)$

$\sim \Omega(n^3) \subset \Omega(n^2)$

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n){  
    if (n <= 1)  
        return 0;  
  
    unsigned sum = desperdicio (n / 2) + desperdicio (n / 2) + desperdicio (n / 2);  
  
    for (unsigned i = 1; i < n-1; i ++)  
        for (unsigned j = 1; j <= i; j ++)  
            for (unsigned k = 1; k <= j; k ++)  
                sum += i*j*k;  
  
    return sum;  
}  
  
=  $\Theta(n^3)$   
~  $\Theta(n^3 \log n)$   
~  $\Theta(3^n)$ 
```

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.

```
~  $\Theta(\log(\sqrt{n})) = \Theta(2 * \log(n))$   
=  $\Theta(\log^2(n)) = \Theta(\log(n))$   
~  $\Theta(\log(n)) = \Theta(\log(n^2))$ 
```

Un problema de tamaño n puede transformarse en tiempo $O(n^3)$ en ocho de tamaño $n/2$; 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?

```
~  $O(n^2 \log n)$   
=  $O(n^3 \log n)$   
~  $O(n^3)$ 
```

¿Cuál es la complejidad temporal de la siguiente función?

```
int f (int n){  
    int k = 0;  
    for (int i = 1; i < n; i *= 3)  
        for (int j = 1; j < i; j += 3)  
            k++;  
    return k;  
}  
= \(\Theta(n)\)  
~  $\Theta(n)$   
~  $\Theta(n \log n)$ 
```

Un vector de enteros de tamaño n tiene sus elementos estructurados en una forma de montículo (heap). ¿Cuál es la complejidad temporal en el peor de los casos de borrar el primer elemento del sector y reconstruirlo posteriormente para que siga manteniendo la estructura del montículo.

```
~ O(n)  
= O(log n)  
~ O(n log n)
```

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?

```
~ Un algoritmo voraz  
= Programación dinámica  
~ Divide y vencerás
```

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.

```
= Si  $f \in \Theta(g)$  entonces  $O(f) = O(g)$   
~ Si  $f \in \Omega(g)$  entonces  $O(f) = \Omega(g)$   
~ Si  $f \in O(g)$  entonces  $g \notin O(f)$ 
```

En un algoritmo de ramificación y poda, ¿Qué ocurre si coinciden los valores obtenidos por las cotas pesimista y optimista del mismo nodo?

~ Esta situación no puede ocurrir en ningún caso; por lo tanto, una de las cotas está mal calculada (o ambas).

= Que ese valor es a su vez el mejor valor que se puede obtener con ese nodo.

~ Que es un nodo hoja, esta situación solo es posible en los nodos hoja.

Di cual de estos tres problemas de optimización no comporta, en el peor caso, tener que considerar $O(n!)$ posibles soluciones.

~ El problema de la asignación de n tareas a n trabajadores de forma que cada trabajador hace exactamente una tarea y cada tarea es asignada a un trabajador exactamente, de forma que la suma de los costes de la tarea es mínimo.

~ El problema del viajante de comercio, 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.

= El 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).

Sea $T(n) = n + T(n - 1)$ para $n > 1$ y $T(1) = 1$. Una de las siguientes afirmaciones siguientes es cierta. Cual?

= $T(n) \in O(n^3)$

~ $T(n) \in O(n \log n)$

~ $T(n) \in O(n)$

Una de las siguientes afirmaciones es falsa. Cual?

= $O(2^n) = O(3^n)$

~ $O(n \log n) \subset O(n^2)$

~ $O(n^2 + 2n + 1) = O(n^2)$

Se pretende resolver el problema del viajante de comercio mediante ramificación y poda. ¿Cuál de las siguientes acciones resulta ser mejor cota optimista para aplicarla a los nodos intermedios?

= Obtener el árbol de recubrimiento de mínimo coste a los vértices aun no visitados.

~ Completar el recorrido pendiente mediante un algoritmo voraz.

~ Asumir que ya no quedan mas vértices por visitar y cerrar el camino desde el ultimo vértice visitado.

Se pretende implementar mediante programación dinámica iterativa la siguiente función recursiva:

```
int f (int m, int n, int p, int * v){  
    if (n < 0) return 0;  
    int aux = 0;  
    if (v[n] <= m)  
        aux = p + f(m - v[n], n - 1, p, v);  
    return aux + f (m, n - 1, p, v);  
}
```

~ $\Theta(m)$

= $\Theta(m * n)$

~ Ninguna de las otras dos opciones es correcta

¿Puede ser que una solución recursiva con memorización 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 siendo más rápida?

~ No: las dos soluciones tienen que realizar el mismo numero de evaluaciones de la función que computa el calor de una solución parcial, y la solución recursiva es más lenta porque tiene que gestionar las llamadas recursivas.

= Si; de hecho, esto pasa en el problema de la mochila discreta con pesos enteros.

~ No: las soluciones recursivas realizan mas evaluaciones de la función que computa el valor de una solución parcial que las iterativas correspondientes.

¿Cual 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; i < 2; j++)  
            count += 1;  
  
    return count;  
}
```

= O(n)

~ O(n log n)

~ O(n^2)

¿Cual de estos conceptos pertenece a la misma categoría que divide y vencerás, vuelta atrás o ramificación y poda?

= programación dinámica

~ memoizacion

~ cota optimista

Se quieren ordenar d numeros distintos comprendidos entre 1 y n. Para ello se usa un array de n booleanos que se inicializan primero a false. A continuacion se recorren los d numeros cambiando los valores del elemento del vector de booleanos correspondiente a su numero a true. Por ultimo se recorre el vector de booleanos escribiendo los indices de los elementos del vector de booleanos que son true. ¿Es este algoritmo mas rapido (asintoticamente) que el mergesort?

- ~ Si, ya que el mergesort es $O(n \log n)$ y este es $O(n)$
- ~ No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.
- = Solo si $d \log d > k n$ (donde k es una constante que depende de la implementacion)

Uno de estos tres problemas no tiene una solucion trivial y eficiente que siga el esquema voraz.

- ~ El problema de la mochila continua.
- = El problema del cambio.
- ~ El problema de la mochila discreta sin limitacion en la carga maxima de la mochila.

En el esquema de vuelta atras el orden en el que se van asignando los distintos valores a las componentes del vector que contendra la solucion...

- ~ ... puede ser relevante si se utilizan mecanismos de poda basados en estimaciones optimistas.
- = Las dos anteriores son ciertas.
- ~ ... es irrelevante si no se utilizan mecanismos de poda basados en la mejor solucion hasta el momento.

```
void f(int n, int arr[])
{
    int i = 0, j = 0;
    for(; i < n; ++i)
        while(j < n && arr[i] < arr[j])
            j++;
}
```

- ~ $O(n \log n)$
- ~ $O(n^2)$
- = $O(n)$

Los algoritmos de vuelta atras que hacen uso de cotas optimistas generan las soluciones posibles al problema mediante . . .

- = . . . un recorrido en profundidad del arbol que representa el espacio de soluciones.
- ~ . . . un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subarboles mas prometedores del espacio de soluciones.
- ~ . . . un recorrido guiado por estimaciones de las mejores ramas del arbol que representa el espacio de soluciones.

Cual de estos problemas tiene una solucion eficiente utilizando programacion dinamica?

- ~ La mochila discreta sin restricciones adicionales.
- = El problema del cambio.
- ~ El problema de la asignacion de tareas.

En un algoritmo de ramificacion y poda, si la lista de nodos vivos no est a ordenada de forma apropiada . . .

- ~ . . . podria ocurrir que se pade el nodo que conduce a la solucion optima.
- = . . . podria ocurrir que se exploren nodos de forma innecesaria.
- ~ . . . podria ocurrir que se descarten nodos factibles.

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 tambien se conocen las coordenadas geograficas de cada ciudad se quiere usar la distancia geografica (en linea recta) entre cada par de ciudades como cota para limitar la busqueda en un algoritmo de vuelta atras. ¿Que tipo de cota ser ia?

- = Una cota optimista.
- ~ Una cota pesimista.
- ~ No se trataria de ninguna poda puesto que es posible que esa heuristica no encuentre una solucion factible.

Cuando se usa un algoritmo voraz para abordar la resolucion de un problema de optimizacion por seleccion discreta (es decir, un problema para el cual la solucion consiste en encontrar un subconjunto del conjunto de elementos que optimiza una determinada funcion), ¿cual de estas tres cosas es imposible que ocurra?

- ~ Que el algoritmo no encuentre ninguna solucion.
- ~ Que la solucion no sea la optima.
- = Que se reconsidera la decision ya tomada anteriormente respecto a la seleccion de un elemento a la vista de la decision que se debe tomar en un instante.

¿Cual de estas estrategias para calcular el n-esimo elemento de la serie de Fibonacci ($f(n) = f(n-1)+f(n-2)$, $f(1) = f(2) = 1$) es mas eficiente?

- ~ Para este problema, las dos estrategias citadas serian similares en cuanto a eficiencia
- = Programacion dinamica
- ~ La estrategia voraz

La siguiente relacion de recurrencia expresa la complejidad de un algoritmo recursivo, donde $g(n)$ es una funcion polinomial: $T(n) = 1$ si $n \leq 1$ // $2T(n/2) + g(n)$ en otro caso Di cu al de las siguientes afirmaciones es falsa:

- ~ Si $g(n)$ pertenece $O(1)$ la relacion de recurrencia representa la complejidad temporal del algoritmo de busqueda dicotomica.
- = Si $g(n)$ pertenece $O(n^2)$ la relacion de recurrencia representa la complejidad temporal del algoritmo de busqueda por insercion.
- ~ Si $g(n)$ pertenece $O(n)$ la relacion de recurrencia representa la complejidad temporal del algoritmo de ordenacion mergesort.

Sea la siguiente relacion de recurrencia $T(n) = 1$ si $n \leq 1$ // $2T(n/2) + g(n)$ en otro caso Si $T(n)$ pertenece $O(n)$, ¿en cual de estos tres casos nos podemos encontrar?

- = $g(n) = 1$
- ~ $g(n) = n^2$
- ~ $g(n) = n$

Si un problema de optimizacion lo es para una funcion que toma valores continuos ...

- ~ La programacion dinamica iterativa siempre es mucho mas eficiente que la programacion dinamica recursiva en cuanto al uso de memoria.
- ~ El uso de memoria de la programacion dinamica iterativa y de la programacion dinamica recursiva es el mismo independientemente de si el dominio es discreto o continuo.
- = La programacion dinamica recursiva puede resultar mucho mas eficiente que la programacion dinamica iterativa en cuanto al uso de memoria.

Estudiad la relacion de recurrencia: $T(n) = 1$ si $n \leq 1$ // $p*T(n/q) + g(n)$ en otro caso (donde p y q son enteros mayores que 1). Di cual de los siguientes esquemas algoritmicos produce de manera natural relaciones de recurrencia asi.

- ~ Programacion dinamica
- = Divide y venceras
- ~ Ramificacion y poda

Cuando se resuelve el problema de la mochila discreta usando la estrategia de vuelta atras, ¿puede ocurrir que se tarde menos en encontrar la solucion optima si se prueba primero a meter cada objeto antes de no meterlo?

- ~ Si, tanto si se usan cotas optimistas para podar el arbol de busqueda como si no.
- ~ No, ya que en cualquier caso se deben explorar todas las soluciones factibles.
- = Si, pero solo si se usan cotas optimistas para podar el arbol de busqueda.

Garantiza el uso de una estrategia “divide y vencer as” la existencia de una solucion de complejidad temporal polinomica a cualquier problema?

- ~ Si, en cualquier caso.
- ~ Si, pero siempre que la complejidad temporal conjunta de las operaciones de descomposicion del problema y la combinacion de las soluciones sea polinomica.
- = No

El algoritmo de ordenacion Quicksort divide el problema en dos subproblemas.¿Cual es la complejidad temporal asintotica de realizar esa division?

- = $O(n)$
- ~ $O(n \log n)$
- ~ mejor(n) y $O(n^2)$

¿Se puede reducir el coste temporal de un algoritmo recursivo almacenando los resultados devueltos por las llamadas recursivas?

- ~ No, solo se puede reducir el coste convirtiendo el algoritmo recursivo en iterativo
- = Si, si se repiten llamadas a la funcion con los mismos argumentos
- ~ No, ello no reduce el coste temporal ya que las llamadas recursivas se deben realizar de cualquier manera

¿Para cual de estos problemas de optimizacion se conoce una solucion voraz?

- ~ El problema de la mochila discreta.
- ~ El problema de la asignacion de coste minimo de n tareas a n trabajadores cuando el coste de asignar la tarea i al trabajador j , c_{ij} esta tabulado en una matriz.
- = El arbol de recubrimiento minimo para un grafo no dirigido con pesos.

Cual de los siguientes criterios proveeria una cota optimista para el problema de encontrar el camino mas corto entre dos ciudades (se supone que el grafo es conexo).

- = Calcular la distancia geometrica (en linea recta) entre la ciudad origen y destino.
- ~ Utilizar la solucion (suboptima) que se obtiene al resolver el problema mediante un algoritmo voraz.
- ~ Calcular la distancia recorrida moviendose al azar por el grafo hasta llegar (por azar) a la ciudad destino.

Decid cual de estas tres es la cota pesimista mas ajustada al valor optimo de la mochila discreta:

- = El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor especifico de los objetos.
- ~ El valor de una mochila que contiene todos los objetos restantes aunque se pase del peso maximo permitido.
- ~ El valor de la mochila continua correspondiente.

La solucion recursiva ingenua (pero correcta) a un problema de optimizacion llama mas de una vez a la funcion con los mismos parametros. Una de las siguientes tres afirmaciones es falsa.

- ~ Se puede mejorar la eficiencia del algoritmo guardando en una tabla el valor devuelto para cada conjunto de parametros de cada llamada cuando esta se produce por primera vez.
- ~ 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.
- = Se puede mejorar la eficiencia del algoritmo convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento basico.

¿Que tienen en comun el algoritmo que obtiene el k-esimo elemento mas pequeño de un vector (estudiado en clase) y el algoritmo de ordenacion Quicksort?

- = La division del problema en subproblemas.
- ~ El numero de llamadas recursivas que se hacen.
- ~ La combinacion de las soluciones a los subproblemas.

¿Cual de los siguientes pares de problemas son equivalentes en cuanto al tipo de solucion (optima, factible, etc.) aportada por el metodo voraz? Seleccione una:

- ~ El fontanero diligente y el problema del cambio.
- = El fontanero diligente y la mochila continua.
- ~ El fontanero diligente y la asignacion de tareas.

En un algoritmo de ramificacion y poda, el orden escogido para priorizar los nodos en la lista de nodos vivos . . .

- ~ . . . determina la complejidad temporal en el peor de los casos del algoritmo.
- = . . . puede influir en el numero de nodos que se descartan sin llegar a expandirlos.
- ~ . . . nunca

Si $f(n)$ pertenece a $O(n^2)$, ¿podemos decir siempre que $f(n)$ pertenece a $O(n^3)$?

- ~ Solo para valores bajos de n
- ~ No, ya que n^2 no pertenece a $O(n^3)$
- = Si ya que n^2 pertenece a $O(n^3)$

Sea $g(n) = (\text{sumatorio [desde } i=0 \text{] hasta } K) \text{ de } a_i * n^i$. Di cual de las siguientes afirmaciones es falsa:

- ~ $g(n)$ pertenece a mejor (n^K)
- = Las otras dos afirmaciones son ambas falsas.
- ~ $g(n)$ pertenece a promedio(n^K)

Sea A una matriz cuadrada $n \times n$. Se trata de buscar una permutacion de las columnas tal que la suma de los elementos de la diagonal de la matriz resultante sea minima. Indicad cual de las siguientes afirmaciones es falsa.

- ~ La complejidad temporal de la mejor solucion posible al problema esta en mejor (n^2).
- ~ Si se construye una solucion al problema basada en el esquema de ramificacion y poda, una buena eleccion de cotas optimistas y pesimistas podria evitar la exploracion de todas las permutaciones posibles.
- = La complejidad temporal de la mejor solucion posible al problema es $O(n \log n)$.

La version de Quicksort que utiliza como pivote el elemento del vector que ocupa la posicion central . . .

- ~ ... se comporta peor cuando el vector ya esta ordenado.
- = ... se comporta mejor cuando el vector ya esta ordenado.
- ~ ... no presenta caso mejor y peor para instancias del mismo tamaño.

Los algoritmos de ordenacion quicksort y mergesort tienen en comun:

- ~ que ordenan el vector sin usar espacio adicional
- ~ que ejecutan en tiempo $O(n)$
- = que aplican la estrategia de Divide y vencerás

¿Cual es la diferencia principal entre una solucion de vuelta atras y una solucion de ramificacion y poda para el problema de la mochila?

- ~ el coste asintotico en el caso peor
- ~ el hecho de que la solucion de ramificacion y poda puede empezar con una solucion suboptima voraz y backtracking no
- = el orden de exploracion de las soluciones

Tenemos un conjunto de n enteros positivos y queremos encontrar el subconjunto de tamaño m y de suma minima.

- ~ lo mas adecuado seria usar una tecnica de ramificacion y poda , aunque en el peor caso el coste temporal asintotico (o complejidad temporal) seria exponencial
- ~ para encontrar la solucion habria que probar con todas las combinaciones posibles de m enteros, con lo que la tecnica de ramificacion y poda no aporta nada con respecto a vuelta atras.
- = una tecnica voraz daria una solucion optima

Di cual de estos resultados de coste temporal asintotico es falsa

- ~ la ordenacion de un vector usando el algoritmo quicksort requiere en el peor caso $\Omega(n^2)$
- = la ordenacion de un vector usando el algoritmo mergesort requiere en el peor caso un tiempo de $\Omega(n^2)$
- ~ la busqueda binaria en un vector ordenado requiere en el peor caso un tiempo en $O(\log n)$

En el problema del viajante de comercio queremos listar todas las soluciones factibles

- ~ lo mas adecuado seria usar una tecnica de ramificacion y poda ya que es muy importante el orden en el que se exploran las soluciones parciales
- = el orden en el que se exploran las soluciones parciales no es relevante, por ello la poda de ramificacion y poda no aporta nada con respecto a la vuelta atras
- ~ lo mas importante es conseguir una cota pesimista muy adecuada . las diferencias entre ramificacion y poda y vuelta atras son irrelevantes en este caso.

la complejidad temporal (o coste temporal asintotico) en el mejor de los casos

= es una funcion de la talla , o tamaño del problema, que tiene que estar definida para todos los posibles valores de esta

~ es el tiempo que tarda el algoritmo en resolver la talla mas pequeña que se le puede presentar

~ las dos anteriores son verdaderas

tenemos n substancias diferentes en polvo y queremos generar todas las distintas formas de mezclarlas de forma que el peso no supere un gramo como la balanza que tenemos solo tiene precision de 0.1 gramos no se consideraran pesos que no sean multipolos de esa cantidad. queremos hacer un programa que genere todas las combinaciones posibles

= no hay ningun problema en usar una tecnica de vuelta atras

~ no se puede usar backtracking porque las decisiones no son valores abstractos

~ no se puede usar backtracking porque el numero de combinaciones es infinito

un algoritmo recursivo basado en el esquema divide y vencerás ...

= ...alcanza su maxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a

~ ... nunca tendra un coste temporal asintotico (o complejidad temporal) asintotico

~ las dos anteriores son verdaderas

dado un problema de optimizacion ¿cuando se puede aplicar el metodo de vuelta atras?

= es condicion necesaria(aunque no suficiente) que el dominio de las decisiones sea discreto o discretizable

~ es condicion necesaria y suficiente que el dominio de las decisiones sea discreto o discretizable

~ no solo es condicion necesaria que el dominio de las decisiones sea discreto o discretizable, ademas debe cumplirse que se puedan emplear mecanismos de poda basados en la mejor solucion hasta el momento.

cual de estas tres expresiones es cierta?

= $O(2^{\log(n)})$ C $O(N^2)$ C $O(2^n)$

~ $O(n^2)$ C $O(2^{\log(n)})$ C $O(2^n)$

~ $O(n^2)$ C $O(2^{\log(n)})$ C $O(2^n)$

sea $f(n)$ la solucion de la relacion de recurrencia $f(n) = 2f(n/2) + n$; $f(1) = 1$ indicad cual de estas tres expresiones es cierta

- ~ $f(n)$ pertenece promedio(n^2)
- = $f(n)$ pertenece promedio($n \log n$)
- ~ $f(n)$ pertenece promedio(n)

indicad cual de estas tres expresiones es falsa

- ~ promedio($n/2$) = promedio(n)
- ~ promedio(n) C O(n)
- = promedio(n) C promedio(n^2)

indica cual es el coste temporal en funcion de n del problema siguiente $s=0$; $\text{for}(i=0; i < n; i++) \text{for } (j=i; j < n; j++) s+=n*ij;$

- ~ es $O(n^2)$ pero no $\Omega(n^2)$
- = es promedio(n^2)
- ~ es promedio(n)

un programa con dos bucles anidados uno dentro de otro , cada uno de los cuales hace aproximadamente n iteraciones , tarda un tiempo

- = $O(n^2)$
- ~ $O(2^n)$
- ~ $O(n)$

la eficiencia de los algoritmos voraces se basa en...

- ~ ... el hecho de que , con antelacion las, posibles decisiones se ordenan de mejor a peor
- = ... el hecho de que las decisiones tomadas no se reconsideran
- ~ ... en el esquema voraz no se puede hablar de eficiencia puesto que a menudo no resuelve el problema

en el esquema de backtracking , los mecanismos de poda basados en la mejor solucion en curso

- = pueden eliminar vectores que representan posibles soluciones factibles
- ~ garantizan que no se va a explotar todo el espacio de soluciones posibles
- ~ las dos anteriores son verdaderas

sea $f(n)$ la solucion de la relacion de recurrencia $f(n) = 2f(n-1) + 1$; $f(1) = 1$. Indicad cual de estas tres expresiones es cierta

- ~ $f(n)$ pertenece promedio(n^2)
- = $f(n)$ pertenece promedio(2^n)
- ~ $f(n)$ pertenece promedio(n)

pertenece $3n^2 + 3$ a $O(n^3)$

- ~ no
- ~ solo para $c=1$ y $n_0 = 5$
- = si

las relaciones de recurrencia

- ~ aparecen solo cuando la solucion es del tipo divide y venceras
- = expresan recursivamente el coste temporal de un algoritmo
- ~ sirven para reducir el coste temporal de una solucion cuando es prohibitivo

el coste temporal de un algoritmo se ajusta a la siguiente ecuacion de recurrencia : $t(n) = 1$ para $n=0, \dots, n$ + sumatorio de $t(j)$ para $j < n$, que coste temporal asintotico o complejidad temporal tendra el algoritmo

- ~ $O(n \log n)$
- ~ $O(n^2)$
- = $O(2^n)$

la version del quicksort que ocupa como pivote el elemnto que ocupa la posicion central

- ~ no presenta caso mejor y peor para instancias del mismo tamaño
- = se comporta mejor cuando el vector ya esta ordenado
- ~ se comporta peor cuando el vector ya esta ordenado

de los problemas siguientes , indicad cual no se puede tratar eficientemente como los otros dos

- = el problema del corte de tubos, que se obtenga el maximo beneficio posible
- ~ el problema del cambio , o sea, el de entregar una cantidad de dinero usando las minimas monedas
- ~ el problema del viajante de comercio

De los problemas siguientes, indicad cual no se puede tratar eficientemente como los otros dos:

- ~ El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el minimo de monedas posibles.
- ~ El problema de cortar un tubo de forma que se obtenga el maximo beneficio posible.
- = El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.

que algoritmo es asintoticamente mas rapido el quicksort o el mergesort?

- ~ como su nombre indica, quicksort
- = son los dos igual de rapidos , ya que el coste temporal asintotico de ambos es $O(n \log n)$
- ~ el mergesort es siempre el mas rapido o igual (salvo una constante) que el quicksort

el coste temporal del algoritmo de ordenacion por insercion es

- = $O(N^2)$
- ~ $O(N)$
- ~ $O(n \log n)$

los algoritmos de programacion dinamica hacen uso

- ~ de que la solucion optima se puede construir anadiendo el componente optimo de los restantes , uno a uno
- = de que se puede ahorrar esfuerzo guardando los resultados de esfuerzos anteriores
- ~ de una estrategia trivial consistente en examinar todas las soluciones posibles

cual de estos tres problemas de optimizacion no tiene una solucion voraz que sea optima

- ~ el problema de la mochila continua o con fraccionamiento
- = el problema de la mochila discreta
- ~ el arbol de cobertura de coste minimo de un grafo conexo

el problema de la funcion compuesta minima consiste en encontrar a partir de un conjunto de funciones dadas , al secuencia minima de composiciones de estas que permita trasformar un numero n en otro m. se quiere resolver mediante ramificacion y poda. cual seria la forma mas adecuada de representar las posibles soluciones?

- = mediante un vector de booleanos
- ~ mediante un vector de reales
- ~ este problema no se puede resolver usando ramificacion y poda si no se fija una cota superior al numero total de aplicaciones de funciones

cual de estas expresiones es falsa?

- = $2n^2 + 3n + 1$ pertenece $O(n^3)$
- ~ $n + n\log n$ pertenece $\Omega(n)$
- ~ $n + n\log n$ pertenece promedio(n)

si el coste temporal de un algoritmo es $T(8n)$, ¿cuál de las siguientes situaciones es imposible?

- ~ $T(n)$ pertenece $O(n)$ y $T(n)$ pertenece promedio(n)
- ~ $T(n)$ pertenece $\Omega(n)$ y $T(n)$ pertenece promedio(n^2)
- = $T(n)$ pertenece promedio(n) y $T(n)$ pertenece $\Omega(n^2)$

el coste temporal asintotico de insertar un elemento en un vector ordenado de forma que continue ordenado es

- = $O(n)$
- ~ $O(\log n)$
- ~ $O(n^2)$

¿que nos proporciona la media entre el coste temporal asintotico(o complejidad temporal) en el peor caso y el coste temporal asintotico en el mejor caso?

- ~ el coste temporal promedio
- ~ el coste temporal asintotico en el caso medio
- = nada de interes

en ausencia de cotas optimistas y pesimistas, la estrategia de backtracking...

- ~ no se puede usar para resolver problemas de optimizacion
- = no recorre todo el arbol si hay manera de descartar subarboles que representen conjuntos de soluciones no factibles
- ~ debe recorrer siempre todo el árbol

el coste temporal asintotico del programa

s = 0; for(i = 0; i < n; i++) for(j = i; j<n;j++) s+= i*j;
y del programa
s = 0; for(i = 0; i < n; i++) for(j = 0; j<n;j++) s+= i*i*j;

son

- ~ el del primero, menor que el segundo
- ~ el del segundo, menor que el primero
- = iguales

se desea obtener todas las permutaciones de una lista compuesta por n elementos

¿Que esquema es el mas adecuado?

- ~ divide y venceras , puesto que la division en sublistas se podria hacer en tiempo constante
- ~ ramificacion y poda , puesto que con buenas funciones de cota es mas eficiente que vuelta atras
- = vuelta atras , es el esquema mas eficiente para este problema

la solucion recursiva ingenua a un determinado problema de optimizacion muestra estas dos caracteristicas: por un lado, se basa en obtener soluciones optimas a problemas parciales mas pequeños y por otro, estos subproblemas se resuelven mas de una vez durante el proceso recursivo. Este problema es candidato a tener una solucion alternativa basada en...

- ~ un algoritmo del estilo de divide y venceras
- = un algoritmo de programacion dinamica
- ~ un algoritmo voraz

decid cual de estas tres es la cota optimista mas ajustada al valor optimo de la mochila discreta

- = el valor de la mochila continua correspondiente
- ~ el valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor especifico de los objetos.
- ~ el valor de una mochila que contiene todos los objetos aunque se pase del peso maximo permitido

la funcion Y de un numero semientero positivo(un numero es semientero si al restarle 0.5 es entero) se define como

```
double gamma(double n)
if(n == 0.5)
return sqrt(PI);
return n*gamma(n-1)
```

se puede calcular usando programacion dinamica iterativa?

= si

~ no, ya que el indice del almacen seria un numero real y no entero

~ no, ya que no podriamos almacenar los resultados intermedios en el almacen

un tubo de n cm de largo se puede cortar en segmentos de 1 centimetro , 2 centimetros 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 que mas ingresos nos producira . se quiere resolver el problema mediante vuelta atras

¿cuál seria la forma mas adecuada de representar las posibles soluciones?

~ un vector de booleanos

~ un par de enteros que indiquen los cortes realizados y el valor acumulado

= una tabla que indique para cada posicion donde se va a cortar cada uno de los posibles valores acumulados

cualquier descomposicion de un problema da lugar a subproblemas de tamaño similar al original, muchos de los cuales se repiten, que esquema es a priori mas apropiado?

~ divide y vencerás

= programacion dinamica

~ ramificacion y poda

un problema de tamaño n puede transformarse en tiempo $O(n^2)$ en n/3 de tamaño n/3 por otro lado la solucion al problema cuando la talla es 1 requiere un tiempo constante, ¿cuál de estas clases de coste temporal asintotico es la mas ajustada?

~ $O(n \log n)$

= $O(n^2 \log n)$

~ $O(n^2)$

indica cual de las siguientes expresiones es falsa

- ~ $\text{promedio}(n/2) = \text{promedio}(n)$
- ~ $\text{promedio}(n) \in O(n)$
- = $\text{promedio}(n) \in O(n^2)$

sea la solucion a la relacion de recurrencia $f(n) = 2f(n/2) + n$ // f(1) = 1

- ~ f(n) pertenece promedio(n^2)
- ~ f(n) pertenece promedio(n)
- = f(n) pertenece promedio(n logn)

para que la complejidad de un algoritmo presente caso mejor y peor distintos

- ~ es condicion necesaria y suficiente que existan instancias distintas del problema con el mismo tamaño
- = es condicion necesaria que existan instancias distintas del problema con el mismo tamaño
- ~ es condicion suficiente que existan instancias distintas del problema con el mismo tamaño

un problema de tamaño n puede transformarse en O(n) en siete de tamaño $n/7$, por otro lado la solucion al problema cuando la talla es 1 requiere tiempo constante ¿ que cota es mas ajustada?

- ~ $O(n^2)$
- ~ $O(n)$
- = $O(n \log n)$

la complejidad temporal en el mejor de los casos de un algoritmo recursivo

- ~ coincide con el valor del caso base de la ecuacion de recurrencia que expresa la complejidad del algoritmo
- = las demás opciones son falsas
- ~ siempre coincidira con la complejidad temporal de las instancias que estan en el caso base del algoritmo recursiva

cual de las siguientes complejidades es la mejor que nos podemos encontrar?

= $O(2^n)$

~ $O(n!)$

~ $O(n^2)$

al resolver el problema del viajante de comercio mediante backtracking asumiendo un grafo de n vertices totalmente conexo cual de estas es una buena cota pesimista al iniciar la busqueda?

- ~ se multiplica n por la distancia de la arista mas corta que nos queda por considerar
- ~ se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las n aristas mas cortas
- = se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vertice mas cercano al ultimo añadido.

se desea obtener todas las permutaciones de una lista compuesta por n elementos ¿que esquema es el mas adecuado?

- ~ ramificacion y poda puesto que con buenas funciones de cota es mas eficiente para este problema que backtracking
- ~ divide y venceras puesto que la division en sublistas se podria hacer en tiempo constante
- = backtracking , para este problema no hay un esquema mas eficiente

la complejidad en el mejor de los casos de un algoritmo de ramificacion y poda

- ~ es siempre exponencial con el numero de decisiones a tomar
- ~ suele ser polinomica con el numero de alternativas por cada decision
- = puede ser polinomica con el numero de decisiones a tomar

la complejidad en el peor de los casos de un algoritmo de ramificacion y poda

- ~ puede ser exponencial con el numero de alternativas por cada decision
- ~ puede ser polinomica con el numero de decisiones a tomar
- = es exponencial con el numero de decisiones a tomar

la estrategia de ramificacion y poda genera las soluciones posibles al problema mediante

= un recorrido guiado por estimaciones de las mejores ramas del arbol que representa el espacio de soluciones

~ un recorrido en profundidad del arbol que representa el espacio de soluciones

~ un recorrido en anchura del arbol que representa el espacio de soluciones

para que sirven las cotas pesimistas en ramificacion y poda

~ para tener la certeza de que la cota optimista esta bien calculada

~ para descartar nodos basandose en la preferencia por algun otro nodo ya completado

= para descartar nodos basandose en el beneficio esperado

en los algoritmos de ramificacion y poda ¿ el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

~ en general si , si se trata de un problema de maximizacion , aunque en ocasiones ambos valores pueden coincidir

= en general si, si se trata de un problema de minimizacion , aunque en ocasiones ambos valores pueden coincidir

~ no , nunca es asi

en los algoritmos de ramificacion y poda , el valor de una cota pesimista es menor que el valor de una cota optimista (entendiendo que ambas cotas se aplican sobre el mismo nodo)

~ en general si , si se trata de un problema de minimizacion , aunque en ocasiones ambos valores pueden coincidir

= en general si , si se trata de un problema de maximizacion , aunque en ocasiones ambos valores pueden coincidir

~ si, siempre es asi

en los algoritmos de ramificacion y poda

~ el uso de cotas pesimistas solo resulta eficaz cuando se dispone de una posible solucion de partida

= una cota optimista es necesariamente un valor insuperable, de no ser asi se podria podar el nodo que conduce a la solucion optima

~ una cota optimista es necesariamente un valor alcanzable , de no ser asi no esta garantizado que se encuentre la solucion optima

Di cual de estos tres algoritmos no es un algoritmo de divide y vencerás

- ~ Quicksort
- ~ Mergesort
- = el algoritmo de Prim

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 cual de las siguientes afirmaciones es falsa

- ~ la complejidad temporal de la mejor solución posible al problema es $O(n!)$
- ~ 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
- = la complejidad temporal de la mejor solución posible al problema es $\Omega(n^2)$

Sea la siguiente relación de recurrencia $T(n) = 1$ si $n \leq 1$ ///[$2T(n/2) + g(n)$ en otro caso Si $T(n)$ pertenece $O(n^2)$, ¿en cual de estos tres casos nos podemos encontrar?

- ~ $g(n) = n$
- = $g(n) = n^2$
- ~ $g(n) = 1$

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz (greedy) que es óptima?

- = el problema de la mochila discreta
- ~ el problema de la mochila continua o con fraccionamiento
- ~ el árbol de cobertura de coste mínimo de un grafo conexo

Un algoritmo recursivo basado en el esquema divide y vencerás...

- ~ ... nunca tendrá una complejidad exponencial
- = ... será más eficiente cuanto más equitativa sea la división en subproblemas
- ~ las dos anteriores son ciertas

Un problema de tamaño n puede transformarse en tiempo $O(n$ cuadrado) 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?

- ~ $O(2^n)$
- = $O(n^3)$
- ~ $O(n^2)$

La complejidad temporal en el mejor de los casos...

- ~ ... es el tiempo que tarda el algoritmo en resolver el problema de tamaño o talla más pequeña que se le puede presentar
- ~ las otras dos opciones son ciertas
- = ... es una función del tamaño o talla del problema que tiene que estar definida para todos los posibles valores de esta

Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

- ~ 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
- ~ se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido
- = 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

Si un problema de optimización lo es para una función que toma valores continuos ...

- ~ la programación dinámica iterativa siempre es mucho más eficiente que la programación dinámica recursiva ¿? en cuanto al uso de memoria
- = la programación dinámica recursiva puede resultar mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria
- ~ 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

La versión de Quicksort que utiliza como pivote el elemento del vector que ocupa la primera posición...

- ~ ... no presenta caso mejor y peor para instancias del mismo tamaño
- ~ ... se comporta mejor cuando el vector ya está ordenado
- = ... se comporta peor cuando el vector ya está ordenado

Si $f(n)$ pertenece $O(n^3)$, ¿puede pasar que $f(n)$ pertenece $O(n^2)$?

- ~ no, porque n al cubo “no incrementa” $O(n)$ al cuadrado)
- ~ solo para valores bajos de n
- = es perfectamente posible, ya que $O(n^2)$ $\leq O(n^3)$

El valor que se obtiene con el metodo voraz para el problema de la mochila discreta es...

- ~ ... una cota inferior para el valor optimo, pero que nunca coincide con este
- ~ ... una cota superior para el valor optimo
- = ... una cota inferior para el valor optimo que a veces puede ser igual a este

Uno de estos tres problemas no tiene una solucion eficiente que siga el esquema de programacion dinamica

- ~ el problema de la mochila discreta
- = el problema de las torres de Hanoi
- ~ 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

La mejor solucion que se conoce para el problema de la mochila continua sigue el esquema...

- = ... divide y venceras
- ~ ... ramificacion y poda
- ~ ... voraz

En los algoritmos de ramificacion y poda...

- ~ una cota optimista es necesariamente un valor alcanzable, de no ser asi no esta garantizado que se encuentre la solucion optima
- = una cota optimista es necesariamente un valor insuperable, de no ser asi se podria podar el nodo que conduce a la solucion optima
- ~ una cota pesimista es el valor que a lo sumo alcanza cualquier nodo factible que no es el optimo

En el esquema de vuelta atras, los mecanismos de poda basados en la mejor solucion hasta el momento...

- ~ ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles
- ~ las otras dos opciones son ciertas
- = ... pueden eliminar soluciones parciales que son factibles

La solucion recursiva ingenua (pero correcta) a un problema de optimizacion llama mas de una vez a la funcion con los mismos parametros. Una de las siguientes tres afirmaciones es falsa

- ~ se puede mejorar la eficiencia del algoritmo guardando en una tabla el valor devuelto para cada conjunto de parametros de cada llamada cuando esta se produce por primera vez
- ~ 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
- = se puede mejorar la eficiencia del algoritmo convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento basico

Cuando se resuelve el problema de la mochila discreta usando la estrategia de vuelta atras, ¿puede ocurrir que se tarde menos en encontrar la solucion optima si se prueba primero a meter cada objeto antes de no meterlo?

- ~ si, tanto si se usan cotas optimistas para podar el arbol de busqueda como si no
- ~ no, ya que en cualquier caso se deben explorar todas las soluciones factibles
- = si, pero solo si se usan cotas optimistas para podar el arbol de busqueda

Cual de los siguientes algoritmos proveeria una cota pesimista para el problema de encontrar el camino mas corto entre dos ciudades (se supone que el grafo es conexo)

- ~ calcular la distancia geometrica (en linea recta) entre la ciudad origen y destino
- ~ para todas las ciudades que son alcanzables en un paso desde la ciudad inicial, sumar la distancia a dicha ciudad y la distancia geometrica hasta la ciudad destino
- = calcular la distancia recorrida moviendose al azar por el grafo hasta llegar (por azar) a la ciudad destino

Decid cual de estas tres es la cota pesimista mas ajustada al valor optimo de la mochila discreta

- ~ el valor de la mochila continua correspondiente
- ~ el valor de una mochila que contiene todos los objetos aunque se pase del peso maximo permitido
- = el valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor especifico de los objetos

La complejidad en el mejor de los casos de un algoritmo de ramificacion y poda...

- ~ ... es siempre exponencial con el numero de decisiones a tomar
- = ... puede ser polinomica con el numero de decisiones a tomar
- ~ ... suele ser polinomica con el numero de alternativas por cada decision

Una de estas tres situaciones no es posible

- ~ $f(n)$ pertenece $O(n)$ y $f(n)$ pertenece $\Omega(1)$
- = $f(n)$ pertenece mejor caso(n cuadrado) y $f(n)$ pertenece $O(n)$
- ~ $f(n)$ pertenece $O(n)$ y $f(n)$ pertenece $O(n$ cuadrado)

En el esquema de vuelta atras el orden en el que se van asignando los distintos valores a las componentes del vector que contendra la solucion...

- ~ ...es irrelevante si no se utilizan mecanismos de poda basados en la mejor solucion hasta el momento
- ~ ... puede ser relevante si se utilizan mecanismos de poda basados en estimaciones optimistas
- = las otras dos opciones son ciertas

En los algoritmos de ramificacion 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)

- ~ no, nunca es asi
- = en general si, si se trata de un problema de minimizacion, aunque en ocasiones ambos valores pueden coincidir
- ~ en general, si, si se trata de un problema de maximizacion , aunque en ocasiones ambos valores pueden coincidir

El uso de funciones de cota en ramificacion y poda...

- ~ ... transforma en polinómicas complejidades que antes eran exponenciales
- ~ ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema
- = ... puede reducir el número de instancias del problema que pertenecen al caso peor

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?

- ~ si, ya que el mergesort es $O(n \log n)$ y este es $O(n)$
- = solo si $d \log d > k n$ (donde k es una constante que depende de la implementación)
- ~ no, ya que este algoritmo ha de recorrer varias veces el vector de booleanos

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?

- ~ cambiamos la función que damos a la cota pesimista
- = el algoritmo puede aprovechar mejor las cotas optimistas
- ~ la comprobación de las soluciones factibles: en ramificación y poda no es necesario puesto que solo genera nodos factibles

En una cuadricula se quiere dibujar el contorno de un cuadrado de n casillas de lado, ¿cuál será la complejidad temporal del mejor algoritmo que pueda existir?

- ~ $O(n^2)$
- = $O(n)$
- ~ $O(\sqrt{n})$

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

- = programación dinámica
- ~ divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño
- ~ el método voraz

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. Tambien se conocen las coordenadas geograficas de cada ciudad y por tanto la distancia geometrica (en linea recta) entre cada par de ciudades. Se pretende acelerar la busqueda de un algoritmo de ramificacion y poda priorizando los nodos vivos (ciudades) que esten a menor distancia geografica de la ciudad objetivo.

- = el nuevo algoritmo no garantiza que vaya a ser mas rapido para todas las instancias del problema posibles
- ~ el nuevo algoritmo siempre sera mas rapido
- ~ esta estrategia no asegura que se obtenga el camino mas corto

La mejora que en general aporta la programacion dinamica frente a la solucion ingenua se consigue gracias al hecho de que...

- ~ ... en la solucion ingenua se resuelve pocas veces un numero relativamente grande de subproblemas distintos
- ~ El numero de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programacion dinamica
- = ... en la solucion ingenua se resuelve muchas veces un numero relativamente pequeno de subproblemas distintos

¿Cual de estos problemas tiene una solucion eficiente utilizando programacion dinamica?

- ~ el problema de la asignacion de tareas
- = el problema del cambio
- ~ la mochila discreta sin restricciones adicionales

Para cual de estos problemas de optimizacion existe una solucion voraz?

- ~ el problema de la mochila discreta
- ~ el problema de la asignacion de coste minimo de n tareas a n trabajadores cuando el coste de asignar la tarea i al trabajador j, c_{ij} esta tabulado en una matriz
- = el arbol de recubrimiento minimo para un grafo no dirigido con pesos

Cuando se usa un algoritmo voraz para abordar la resolucion de un problema de optimizacion por seleccion discreta (es decir, un problema para el cual la solucion consiste en encontrar un subconjunto del conjunto de elementos que optimiza una determinada funcion), ¿cuál de estas tres cosas es imposible que ocurra?

- ~ que el algoritmo no encuentre ninguna solucion
- = que se considere la decision ya tomada anteriormente respecto a la seleccion de un elemento a la vista de la decision que se debe tomar en el instante actual
- ~ que la solucion no sea la optima

Dado un problema de optimizacion, el metodo voraz.....

- ~ ...siempre obtiene la solucion optima
- = ... garantiza la solucion optima solo para determinados problemas
- ~ ... siempre obtiene una solucion factible

Dado un problema de optimizacion cualquiera, ¿la estrategia de vuelta atras garantiza la solucion optima?

- ~ si, puesto que este metodo analiza todas las posibilidades
- ~ si, siempre que el dominio de las decisiones sea discreto o discretizable y ademas se empleen mecanismos de poda basados en la mejor solucion hasta el momento
- = es condicion necesaria que el dominio de las decisiones sea discreto o discretizable y que el numero de decisiones a tomar este acotado

¿Garantiza el uso de una estrategia “divide y vencerás” la existencia de una solucion de complejidad temporal polinomica a cualquier problema?

- ~ si, en cualquier caso
- = no
- ~ si, pero siempre que la complejidad temporal conjunta de las operaciones de descomposicion del problema y la combinacion de las soluciones sea polinomica

En un problema de optimizacion, si el dominio de las decisiones es un conjunto infinito

- = una estrategia voraz puede ser la unica alternativa
- ~ es probable que a traves de programacion dinamica se obtenga un algoritmo eficaz que lo solucione
- ~ podremos aplicar el esquema vuelta atras siempre que se trate de un conjunto infinito numerable

dado un problema de optimizacion cualquiera¿ la estrategia de backtracking garantiza la solucion optima?

- ~ si, puesto que ese metodo analiza todas las posibilidades
- ~ si , siempre que el dominio de las decisiones sea discreto o discretizable y ademas se empleen mecanismos de poda basados en la mejor solucion hasta el momento
- = es condicion necesaria que el dominio de las decisiones sea discreto o discretizable y que el numero de decisiones a tomar este acotado

en los algoritmos de ramificacion y poda..

- ~ una cota optimista es necesariamente un valor alcanzable, de no ser asi no esta garantizado que se encuentre la solucion optima
- = una cota optimista es necesariamente un valor insuperable , de no ser asi se podria podar el nodo que conduce a la solucion optima
- ~ una cota pesimista es el valor que a lo sumo alcanza cualquier nodo factible que no es el optimo

el uso de funciones de cota en ramificacion y poda

- ~ transforma en polinomicas complejidades que antes eran exponenciales
- ~ garantiza que el algoritmo va a ser mas eficiente ante cualquier instancia del problema
- = puede reducir el numero de instancias del problema que pertenecen al caso peor

la solucion recursiva ingenua(pero correcta) a un problema de optimizacion llama mas de una vez a la funcion con los mismos parametros . una de las siguientes afirmaciones es falsa

- ~ se puede mejorar la eficiencia del algoritmo guardando en una tabla el valor devuelto para cada conjunto de parametros de cada llamada cuando esta se produce por primera vez
- ~ 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
- = se puede mejorar la eficiencia del algoritmo convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento basico

la mejor solucion que se conoce para el problema de la mochila continua sigue el esquema

- = divide y vencerás
- ~ ramificacion y poda
- ~ voraz

si un problema de optimizacion lo es para una funcion que toma valores continuos

- ~ la programacion dinamica iterativa siempre es mucho mas eficiente que la programacion dinamica iterativa en cuanto al uso de memoria
- = la programacion dinamica recursiva puede resultar mucho mas eficiente que la programacion dinamica iterativa en cuanto al uso de memoria
- ~ el uso de memoria de la programacion dinamica iterativa y de la programacion dinamica recursiva es el mismo independientemente de si el dominio es discreto o continuo

al resolver el problema del viajante de comercio mediante backtracking , cual de estas cotas optimistas se espera que pade mejor el arbol de busqueda?

- ~ se multiplica k por la distancia de la arista mas corta que nos queda por considerar donde k es el numero de saltos que nos quedan por dar
- ~ se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vertice mas cercano al ultimo añadido
- = se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las k aristas mas cortas, donde k es el numero de saltos que nos quedan por dar

para cual de estos problemas de optimizacion existe una solucion voraz?

- ~ el problema de la mochila discreta
- ~ el problema de la asignacion de coste minimo de n tareas a n trabajadores cuando el coste de asignar la tarea i al trabajador j c_{ij} esta tabulado en una matriz
- = el arbol de recubrimiento minimo para un grafo no dirigido con pesos

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 tambien se conocen las coordenadas geograficas de cada ciudad y por tanto la distancia geometrica en linea recta entre cada par de ciudades. se pretende acelerar la busqueda de un algoritmo de ramificacion y poda priorizando los nodos vivos(ciudades(que esten a menor distancia geografica de la ciudad objetivo

= el nuevo algoritmo no garantiza que vaya a ser mas rapido para todas las instancias del problema posibles

- ~ el nuevo algoritmo siempre sera mas rapido
- ~ esta estrategia no asegura que se obtenga el camino mas corto

la complejidad temporal en el mejor de los casos

- ~ es el tiempo que tarda el algoritmo en resolver el problema de tamaño o talla mas pequeña que se le puede presentar
 - ~ las otras dos opciones son ciertas
- = es una función del tamaño o talla del problema que tiene que estar definida para todos los posibles valores de esta

la complejidad en el mejor de los casos de un algoritmo de ramificacion y poda

- ~ es siempre exponencial con el numero de decisiones a tomar
- = puede ser polinomial con el numero de decisiones a tomar
- ~ suele ser polinomial con el numero de alternativas por cada decisión

un algoritmo recursivo basado en divide y vencerás

- ~ nunca tendrá una complejidad exponencial
- = será más eficiente cuanto más equitativa sea la división en subproblemas
- ~ las dos anteriores son correctas

cuando se usa un algoritmo voraz para abordar la resolución de un problema de optimización por selección directa(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?

- ~ que el algoritmo no encuentre ninguna solución
- = que se considere 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
- ~ que la solución no sea la óptima

cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, que esquema promete ser más apropiado?

- = programación dinámica
- ~ divide y vencerás , siempre que se garantice que los subproblemas no son del mismo tamaño
 - ~ voraz

sea la siguiente relación de recurrencia

$t(n) + 1$ si $n \leq 1$ // $+ 2T(n/2) + g(n)$ en otro caso.. Si $t(n)$ pertenece $O(n^2)$ en cual de los casos nos podemos encontrar?

~ $g(n) = n$

= $g(n) = n^2$

~ $g(n) = 1$

en el esquema de backtracking los mecanismos de poda basados en la mejor solución hasta el momento

- ~ garantizan que no se va a explotar nunca todo el espacio de soluciones
- ~ las otras dos opciones son ciertas
- = pueden eliminar soluciones parciales que son factibles

uno de estos tres problemas no tiene una solución eficiente que siga el esquema de programación dinámica

~ el problema de la mochila discreta

= el problema de las torres de hanói

~ 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

el valor que se obtiene con el método voraz para el problema de la mochila discreta es

- ~ una cota inferior para el valor óptimo, pero que nunca coincide con este
- ~ una cota superior para el valor óptimo
- = una cota inferior para el valor óptimo que a veces puede ser igual a este

decid cual de estas tres es la cota pesimista mas ajustada al valor óptimo de la mochila discreta

~ el valor de la mochila continua correspondiente

~ el valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido

= el valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos

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?

~ $O(2^n)$

= $O(n^3)$

~ $O(n^2)$

Sea la siguiente relación de recurrencia :::: $T(n) = 1 \text{ si } n \leq 1$ ////////////// $[2T(n/2) + g(n)]$ en otro caso. Si $T(n) \in O(n)$, ¿en cuál de estos tres casos nos podemos encontrar?

~ $g(n) = n^2$

= las otras dos opciones son ambas ciertas

~ $g(n) = \log n$

¿Qué se deduce de $f(n)$ y $g(n)$ se se cumple $\lim_{n \rightarrow \infty} [f(n)/g(n)] = K$, con K distinto de 0?

~ $g(n)$ pertenece $O[O(f(n))]$ pero $f(n)$ "no incremento" $O[g(n)]$

= $f(n)$ pertenece $O[g(n)]$ y $g(n)$ "incremento" $O[f(n)]$

~ $f(n)$ pertenece $O[g(n)]$ pero $g(n)$ "no incremento" $O[f(n)]$

¿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);
}
```

~ promedio($n \times m$)

= promedio(n)

~ promedio (n^2)

Dado un problema de maximizacion resuelto mediante un esquema de ramificacion y poda, ¿que ocurre si la cota optimista resulta ser un valor excesivamente elevado?

- = que se podria explorar mas nodos de los necesarios
- ~ que se podria explorar menos nodos de los necesarios
- ~ que se podria podar el nodo que conduce a la solucion optima

Se desea resolver el problema de la potencia enesima (x^n), asumiendo que n es par y que se utilizara la siguiente recurrencia: $\text{pot}(x,n) = \text{pot}(x,n/2) * \text{pot}(x,n/2)$; ¿Que esquema resulta ser mas eficiente en cuanto al coste temporal?

- ~ en este caso tanto programacion dinamica como divide y venceras, resultan ser equivalentes en cuanto a la complejidad temporal
- = programacion dinamica
- ~ divide y venceras

Dado el problema de las torres de Hanoi resuelto mediante divide y venceras ¿cuál de las siguientes relaciones recurrencia expresa mejor su complejidad temporal para el caso general, siendo n el numero de discos?

- ~ $T(n) = T(n-1) + n$
- = $T(n) = 2T(n-1) + 1$
- ~ $T(n) = 2T(n-1) + n$

El coste temporal asintotico de insertar un elemento en un vector ordenado de forma que continue ordenado es...

- ~ ... $O(\log n)$
- ~ ... mejor(n^2)
- = ... $O(n)$

Dado el problema del laberinto con tres movimientos, se desea saber el numero de caminos distintos desde la casilla inicial (1,1) hasta la casilla (n,m) y para ello se aplica un esquema de divide y venceras, ¿Cual seria la recurrencia apropiada para el caso general?

- ~ ninguna de las otras dos recurrencias se corresponde con un esquema de divide y venceras
- = $nc(n,m) = nc(n-1,m) + nc(n,m-1) + nc(n-1,m-1)$
- ~ $nc(n,m) = nc(n-1,m) * nc(n,m-1) * nc(n-1,m-1)$

Dado un problema de minimizacion resuelto mediante un esquema de ramificacion y poda, ¿que propiedad cumple una cota optimista?

- = las otras dos opciones son ambas faltas
- ~ asegura un ahorro en la comprobacion de todas las soluciones factibles
- ~ siempre es mayor o igual que la mejor solucion posible alcanzada

En el esquema de ramificacion y poda, ¿que estructura es la mas adecuada si queremos realizar una exploracion por niveles?

- = cola
- ~ cola de prioridad
- ~ pila

Dado el problema del laberinto con tres movimientos, ¿se puede aplicar un esquema de programacion dinamica para obtener un camino de salida?

- ~ no, para garantizar que se encuentra un camino de salida hay que aplicar metodos de busqueda exhaustiva como vuelta atras o ramificacion y poda.
- ~ no, con este esquema se puede conocer el numero total de caminos distintos que conducen a la salida pero no se puede saber la composicion de ninguno de ellos.
- = si, en caso de existir con este esquema siempre se puede encontrar un camino de salida

¿Que ocurre si la cota pesimista de un nodo se corresponde con una solucion que no es factible?

- = que el algoritmo seria incorrecto pues podria descartarse un nodo que conduce a la solucion optima.
- ~ que el algoritmo seria mas lento pues se explorarian mas nodos de los necesarios.
- ~ nada especial, las cotas pesimistas no tienen por que corresponderse con soluciones factibles

Se desea ordenar una lista enlazada de n elementos haciendo uso del algoritmo Mergesort. En este caso, al tratarse de una lista, la complejidad temporal asintotica de realizar la division en subproblemas resulta ser lineal con el tamaño de esa lista. ¿Cual seria entonces el coste temporal de realizar dicha ordenacion?

- ~ ninguna de las otras dos opciones es cierta
- = promedio($n \log n$)
- ~ promedio(n^2)

Una de las practicas de laboratorio consistio en el calculo empirico de la complejidad temporal promedio del algoritmo de ordenacion de vectores Quicksort tomando como centinela el elemento del vector que ocupa la posicion central. ¿Cual es el orden de complejidad que se obtuvo?

- = $n \log n$
- ~ $n \log^2 n$
- ~ n^2

Un tubo de n centimetros de largo se puede cortar en segmentos de 1 ctm., 2 ctm, 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 mas ingresos nos producira. Se quiere resolver el problema mediante vuelta atras ¿cuál seria la forma mas adecuada de representar las posibles soluciones?

- ~ un par de enteros que indiquen los cortes realizados y el valor acumulado
- ~ una tabla que indique, para cada posicion donde se va a cortar, cada uno de los posibles valores acumulados
- = un vector de booleanos

Si f pertenece $\text{mejor}(g_1)$ y f pertenece $\text{mejor}(g_2)$ entonces

- ~ f pertenece $\text{mejor}(g_1 * g_2)$
- = f pertenece $\text{mejor}(g_1 + g_2)$
- ~ f no pertenece $\text{mejor}[\min(g_1, g_2)]$

El esquema de vuelta atras...

- = garantiza que encuentra la solucion optima a cualquier problema de seleccion discreta
- ~ se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado
- ~ las otras dos opciones son ambas verdaderas

Que estrategia de busqueda es a priori mas apropiada en un esquema de vuelta atras?

- ~ explorar primero los nodos con mejor cota optimista
- ~ explorar primero los nodos que estan mas completados
- = en el esquema de vuelta atras no se pueden definir estrategias de busqueda

Que complejidad se obtiene a partir de la relacion de recurrencia $T(n) = 8T(n/2) + n^3$ con $T(1) = O(1)$?

- ~ $O(n \log n)$
- = $O(n^3 \log n)$
- ~ $O(n^3)$

Dada la siguiente funcion:

```
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 asintotica?

- ~ $O(n \log n)$
- ~ $O(n "al cuadrado")$
- = $O(n)$

¿Que nos proporciona la media entre el coste temporal asintotico (o complejidad temporal) en el peor caso y el coste temporal asintotico en el mejor caso?

- ~ el coste temporal asintotico en el caso medio
- = nada de interes
- ~ el coste temporal promedio

Si el coste temporal de un algoritmo es $T(n)$, ¿cuál de las siguientes situaciones es imposible?

- ~ $T(n)$ pertenece mejor(n) y $T(n)$ pertenece promedio(n^2)
- = $T(n)$ pertenece promedio(n) y $T(n)$ pertenece mejor(n^2)
- ~ $T(n)$ pertenece $O(n)$ y $T(n)$ pertenece promedio(n)

¿En ramificacion y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?

- = Si, aunque no es una garantía de que sea una buena estrategia de búsqueda
- ~ si, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista
- ~ no, la cota optimista solo se utiliza para determinar si una n-tupla es prometedora

Un algoritmo recursivo basado en el esquema divide y vencerás...

- ~ ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial
- ~ las otras dos opciones son ambas verdaderas
- = ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en " a " problemas de tamaño n/a

Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1,1)$ hasta la casilla (n,m) y para ello se aplica el esquema programación dinámica para obtener un algoritmo lo más eficiente posible en cuanto a complejidad temporal y espacial, ¿cuáles serían ambas complejidades?

- ~ temporal promedio $[\max(n,m)]$ y espacial promedio $[\max(n,m)]$
- ~ temporal promedio $(n \times m)$ y espacial promedio $(n \times m)$
- = temporal promedio $(n \times m)$ y espacial promedio $[\min(n,m)]$

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?

- ~ que en realidad no es un algoritmo voraz pues las decisiones que se toman no deberían reconsiderarse
- ~ que es un algoritmo voraz pero sin garantía de solucionar el problema
- = que en realidad no es un algoritmo voraz pues el criterio de selección no lo e

En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...

- = ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles
- ~ ... debe recorrer siempre todo el árbol
- ~ ... no se puede usar para resolver problemas de optimización

De las siguientes afirmaciones marca la que es verdadera

- = en un esquema de vuelta atras, las cotas pesimistas no tienen sentido si lo que se pretende es obtener todas las soluciones factibles
- ~ el esquema de vuelta atras no es compatible con el uso conjunto de cotas pesimistas y optimistas
- ~ las cotas pesimistas no son compatibles con un esquema de vuelta atras

El esquema voraz...

- ~ puede que no encuentre una solución pero si lo hace se garantiza que es óptima
- = las otras dos opciones son ambas falsas
- ~ garantiza encontrar una solución a cualquier problema, aunque puede que no sea óptima

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?

- = programación dinámica
- ~ ramificación y poda
- ~ divide y vencerás

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?

- ~ de una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización
- ~ 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
- = de una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización

En el esquema de vuelta atras, los mecanismos de poda basados en la mejor solución hasta el momento...

- ~ ... garantizan que no se va a explorar todo el espacio de soluciones posibles
- ~ las otras dos opciones son ambas verdaderas
- = ... pueden eliminar vectores que representan posibles soluciones factibles

Decid cual de estas tres es la cota optimista mas ajustada al valor optimo de la mochila discreta:

- ~ el valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor especifico de los objetos
- ~ el valor de una mochila que contiene todos los objetos aunque se pase del peso maximo
- = el valor de la mochila continua correspondiente

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

- ~ $O(2^{\log n})$ C $O(n^2)$
- = promedio(n) C promedio(n^2)
- ~ $n + n \log n$ pertenece mejor(n)

Dado el problema del laberinto con tres movimientos, ¿cuál de las estrategias siguientes proveeria de una cota optimista para ramificacion y poda?

- ~ suponer que en adelante todas las casillas del laberinto son accesibles
- ~ suponer que ya no se van a realizar mas movimientos
- = las otras dos estrategias son ambas validas

En el problema del viajante de comercio (travelling salesman problem) queremos listar todas las soluciones factibles

- = el orden en el que se exploran las soluciones parciales no es relevante; por ello, la tecnica ramificacion y poda no aporta nada fcon respecto a vuelta atras
- ~ lo mas importante es conseguir una cota pesimista adecuada. Las diferencias entre ramificacion y poda y vuelta atras son irrelevantes en este caso
- ~ lo mas adecuado seria usar una tecnica de ramificacion y poda ya que es muy importante el orden en el que se exploran las soluciones parciales

Se desea obtener todas las permutaciones de una lista compuesta por n elementos. ¿Que esquema es el mas adecuado?

- ~ ramificacion y poda, puesto que con buenas funciones de cota es mas eficiente que vuelta atras
- = vuelta atras, es el esquema mas eficiente para este problema
- ~ divide y vencerás, puesto que la division en sublistas se podria hacer en tiempo constante

¿Cual es la complejidad temporal, en el peor de los casos, del mejor algoritmo que se puede escribir para resolver el problema de la mochila discreta?

- = Exponencial con el numero de objetos a tratar.
- ~ Polinomica con el numero de objetos a tratar, siempre que se utilice programacion dinamica.
- ~ Ninguna de las otras dos son ciertas.

Un algoritmo que calcula una funcion recursivamente tiene coste prohibitivo y se decide mejorararlo transformandolo en un algoritmo de programacion dinamica iterativa, pero se le añade memoizacion. ¿podria ser que el algoritmo iterativo p evaluar la funcion mas veces que el recursivo con memoizacion?

- ~ No, ambos evaluan la funcion el mismo numero de veces.
- ~ No, el recursivo evalua la funcion muchas mas veces.
- = Podria ser, por ejemplo, como ocurre en el caso de la mochila discreta con pesos enteros.

Asumiendo que n es par, las siguientes recurrencias matematicas, obtienen el valor de la potencia enésima (x^n), cual de las siguientes afirmaciones es cierta

- = La primera recurrencia resultara ser la mas eficiente siempre que se utilice la programacion dinamica recursiva para su implementacion.
- ~ Ambas recurrencias son equivalentes en cuanto a complejidad temporal.
- ~ La segunda recurrencia resulta ser la mas eficiente siempre que se utilice divide y venceras.

Con respecto al tamaño del problema ¿Cual es el orden de complejidad temporal asintotica de la siguiente funcion? void traspuesta(mat & A)

- ~ constante
- = lineal
- ~ cuadratico

Si f no pertenece $O(g_1)$ y f pertenece a $O(g_2)$ entonces NO siempre se cumplira

- ~ f pertenece $\Omega(\min(g_1, g_2))$
- ~ f pertenece $O(\max(g_1, g_2))$
- = f pertenece $\Omega(g_1 + g_2)$

Que tiene que valer k en la relacion de recurrencia $T(n) = 1 \text{ } n < 1 // / / / n^k + 2T(n/2)$
 $n > 1$
para que $T(n) = n \log(n)$?

- ~ 0
- = 1
- ~ $\log(n)$

Cual de las siguientes relaciones de recurrencia es la del algoritmo mergesort

- = $T(n) = n + 2T(n/2)$ para $n > 1$
- ~ $T(n) = n + T(n/2)$ para $n > 1$
- ~ $T(n) = n + T(n-1)$ para $n > 1$

Los algoritmos de ordenacion quicksort y mergesort

- = tienen el mismo coste temporal asintotico en el caso mejor
- ~ tienen el mismo coste temporal asintotico en el caso peor
- ~ tienen el mismo coste temporal en los dos casos

Que tiene que valer b en la relacion de recurrencia $T(n) = 1 \text{ } n < 1 // / 1+bT(n-1) \text{ } n > 1$
para que $T(n) = 2^n$

- ~ 1
- = 2
- ~ 0

Queremos resolver por ramificacion y poda el problema de la mochila discreta. Si resolvemos el mismo problema de la forma voraz pero sin ordenar previamente los objetos por valor/peso, obtendremos

- = Una cota pesimista
- ~ Una cota optimista.
- ~ Nada que podamos utilizar

Queremos resolver por vuelta atras el problema de las n reinas. El usar una buena cota optimista permitiria:

- = No es aplicable ese tipo de podas a este problema.
- ~ Muy probablemente, hacer que el programa vaya mas lento.
- ~ Muy probablemente, resolver el problema de forma mas rapida.

Sea V el conjunto de todos los valores faciales que presentan las monedas de un pais, una cantidad M ¿Cual de las siguientes afirmaciones es falsa?

- = El algoritmo que calcularia $n(M)$ asi seria un algoritmo voraz y tendria un coste razonable.
- ~ El algoritmo recursivo que calcularia $n(M)$ asi tendria un coste prohibitivo
- ~ El algoritmo recursivo que calcularia $n(M)$ se podria convertir en un algoritmo con coste razonable usando memoizacion.

El arbol de expansion de minimo coste de un grafo

- = ...puede utilizarse como cota optimista para resolver el problema del viajante de comercio
- ~ ...puede utilizarse como cota pesimista para resolver el problema del viajante de comercio
- ~ Ninguna de las otras dos opciones es verdadera

Si $\lim_{n \rightarrow \infty} g(n)/f(n)$ resulta ser una constante positiva no nula, cual de las siguientes expresiones no puede darse

- ~ $f(n)$ perteneciente $O(g(n))$
- = $g(n)$ no pertenece a $O(f(n))$
- ~ $f(n)$ perteneciente a $\Omega(g(n))$ y $g(n)$

Tenemos un vector ordenado de tamaño n_o y un vector desordenado de tamaño n_d queremos obtener un vector ordenado con todos los elementos ¿Que sera mas rapido?

- = Ordenar el desordenado y luego mezclar las listas.
- ~ Insertar los elementos del vector desordenado (uno a uno) en el vector ordenado.
- ~ Depende de si $n_o > n_d$ o no

Tenemos una lista ordenada de tamaño n_o y una lista desordenada de tamaño n_d , queremos obtener una lista ordenada con todos los elementos, ¿Cual seria la complejidad de insertar uno a uno todos los elementos de la lista desordenada en la ordenada?

- ~ $O(n_d \log n_o)$
- = $O(n_o \times n_d + n_d^2)$
- ~ $O(n_d \times n_o)$

Tenemos una lista recursiva con la siguiente cabecera: double f(const double &) Con solo esta informacion, cual podria ser la definicion adecuada para el almacen?

= Ninguna de las dos otras opciones son verdaderas

~ vector <double> A

~ int A[]

Queremos resolver por ramificacion y poda el problema de la mochila discreta. Si resolvemos el mismo problema de la forma voraz PERMITIENDO COGER OBJETOS FRACCIONADOS pero sin ordenar previamente los objetos por valor/peso, obtendremos

= Nada que podamos utilizar

~ Una cota pesimista

~ Una cota optimista

Cual es el coste espacial asintotico del siguiente algoritmo:

int f(int n)

int a = 1 , r= 0;

for(int i =0 , i<n , i++)

r= a + r;

a= 2*r;

= O(1)

~ O(log(n))

~ O(n)

Que diferencia (entre otras) hay entre el algoritmo de Prim y el de Kruskal?

= El subgrafo que paso a paso va generando el algoritmo de prim siempre contiene una única componente conexa.

~ El algoritmo de Prim es voraz y el de Kruskal no.

~ Aun siendo el grafo de partida totalmente conexo, el algoritmo de Kruskal garantiza la solucion optima mientras que el de prim solo garantiza un sub optimo.

La siguiente relacion de recurrencia expresa la complejidad de un algoritmo recursivo, donde g(n) es una funcion polinomica

Cual es la definicion correcta de Omega(f)

~ O(g) = f: N-> g(n)<cf(n)

= O(g) = f: N-> f(n)<cg(n)

~ O(g) = f: N-> g(n)<cf(n)

¿Que aporta la tecnica de ramificacion y poda frente a vuelta atras?

- ~ Eficiencia, los algoritmos de ramificacion y poda son mas eficaces que los de vuelta atras.
- = La posibilidad de analizar distintas estrategias para seleccionar el siguiente nodo a expandir. YO DIRIA QUE ES ESTA
- ~ La posibilidad de combinar el uso de cotas pesimistas y optimistas para cualquier nodo ya sea completado o sin completar. En vuelta atras esto no se puede hacer. ESTA TAMBIEN PUEDE SER.

Sea n el numero de elementos que contienen los vectores w y v , ¿cuál es la complejidad temporal asintotica en función de n asumiendo que la llamada inicial i toma valor n ?

`float f (vector <float> &w, vector <unsigned> &v)`

- = omega(n) y $O(n^2)$
- ~ promedio(2^n)
- ~ omega(n) y $O(2^n)$

En el siguiente problema de cortar un tubo de longitud n en segmentos de longitud entera entre 1 y n ¿Que debería ir en lugar de XXXXXX?

`void fill(price m[]) +cutrod(XXXXXX)`

- ~ $n, m[n]-1, p$
- = $n-i, m, p$
- ~ $n-m[n], m, p$

Se pretende resolver un problema de maximización utilizando ramificación y poda, y para ello se dispone de 3 cuotas optimistas ¿Cuál deberíamos utilizar para reducir la cantidad de nodos a explorar?

- ~ La que obtenga valores más elevados. PUEDE SER ESTA
- = La que obtenga valores más pequeños. YO PONDRIA ESTA
- ~ La que se acerque más a una heurística voraz que obtenga una solución aproximada.

Tenemos un vector ordenado y queremos comprobar si contiene un elemento dado ¿Cuál será la complejidad temporal más ajustada para hacerlo ?

- ~ El tamaño del vector
- ~ Constante con el tamaño del vector
- = El logaritmo del tamaño del vector

Dadas las siguientes funciones:

//Precondicion: $0 \leq i < v.size(); i < j \leq v.size()$

Se quiere reducir la complejidad temporal usando programación dinámica iterativa, cual sería la complejidad espacial

- ~ cubica
- = cuadratica
- ~ exponencial

Tratándose de un esquema general para resolver problemas de minimización ¿ que falta en el hueco?

Solution BB (Problem p)

if(?????????????)

- = `n.optimistic_b() <= pb`
- ~ `n.optimistic_b() >= pb`
- ~ `n.pesimistic_b() <= pb`

Tratándose de un esquema general para resolver problemas de maximización ¿ que falta en el hueco?

Solution BB (Problem p)

if(?????????????)

- ~ `n.optimistic_b() <= pb`
- = `n.optimistic_b() >= pb`
- ~ `n.pesimistic_b() <= pb`

La programación dinámica...

= Las otras dos opciones son ciertas. Correcta

- ~ ... normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores.
- ~ ... en algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos.

¿Cuál es la mejor complejidad espacial que se puede conseguir?

- = $O(y)$
- ~ $O(y^2)$
- ~ $O(1)$

Se pretende implementar mediante programacion dinamica iterativa la funcion recursiva:

```
float f(unsigned x, int y){  
    if( y < 0 ) return 0;  
    float A = 0.0;  
    if ( v1[y] <= x )  
        A = v2[y] + f( x-v1[y], y-1 );  
    float B = f( x, y-1 );  
    return min(A,2+B);
```

Un informatico quiere subir a una montana y para ello decide que tras cada paso, el siguiente debe tomarlo en la direccion de maxima pendiente hacia arriba. Ademas, entendera que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna direccion que sea cuesta arriba. ¿que tipo de algoritmo esta usando nuestro informatico?

- ~ un algoritmo de programacion dinamica.
- = un algoritmo voraz.
- ~ un algoritmo divide y venceras.

En la solucion al problema de la mochila continua ¿por que es conveniente la ordenacion previa de los objetos?

= Para reducir la complejidad temporal en la toma de cada decision: de $O(n)$ a $O(1)$, donde n es el numero de objetos a considerar.

Correcta

- ~ Para reducir la complejidad temporal en la toma de cada decision: de $O(n^2)$ a $O(n \log n)$, donde n es el numero de objetos a considerar.
- ~ Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.

¿ Como se veria afectada la solucion voraz al problema de la asignacion de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores ?

- ~ Ya no se garantizaria la solucion optima pero si una factible.
- = La solucion factible ya no estaria garantizada, es decir, pudiera ser que el algoritmo no llegue a solucion alguna.
- ~ Habria que replantearse el criterio de seleccion para comenzar por aquellos trabajadores con mas restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solucion factible.

Un tubo de n centimetros de largo se puede cortar en segmentos de 1 centimetro, 2 centimetros, 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 mas ingresos nos producira. Di cual de estas tres afirmaciones es falsa Seleccione una:

- = Hacer una evaluacion exhaustiva de "fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo $\Theta(n!)$.
- ~ Es posible evitar hacer la evaluacion exhaustiva ``de fuerza bruta'' guardando, para cada posible longitud $j < n$ el precio mas elevado posible que se puede obtener dividiendo el tubo correspondiente.
- ~ Hacer una evaluacion exhaustiva ``de fuerza bruta'' de todas las posibles maneras de cortar el tubo consume un tiempo $\Theta(2n)$.

Se pretende implementar mediante programacion dinamica iterativa la funcion recursiva:

```
unsigned f( unsigned y, unsigned x) // suponemos y >= x
if (x==0 || y==x) return 1;
return f(y-1, x-1) + f(y-1, x);
```

¿Cual es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- ~ $O(y^2)$
- = $O(y)$
- ~ $O(1)$

¿Cual de estas estrategias voraces obtiene siempre un mejor valor para la mochila discreta? Seleccione una:

- ~ Meter primero los elementos de mayor valor.
- ~ Meter primero los elementos de mayor valor especifico o valor por unidad de peso.
- = Ninguna de las otras dos opciones es cierta. Correcta

En el metodo voraz Seleccione una:

- ~ ... siempre se encuentra solucion pero puede que no sea la optima.
- = ... es habitual preparar los datos para disminuir el coste temporal de la funcion que determina cual es la siguiente decision a tomar.
- ~ ... el dominio de las decisiones solo pueden ser conjuntos discretos o discretizables.

¿Que mecanismo se usa para acelerar el algoritmo de Prim? Seleccione una:

- ~ El TAD "Union-find"
- ~ Mantener una lista de los arcos ordenados segun su peso.
- = Mantener para cada vertice su "padre" mas cercano.

En la solucion al problema de la mochila continua ¿por que es conveniente la ordenacion previa de los objetos? Seleccione una:

- = Para reducir la complejidad temporal en la toma de cada decision: de $O(n)$ a $O(1)$, donde n es el numero de objetos a considerar.
- ~Para reducir la complejidad temporal en la toma de cada decision: de $O(n^2)$ a $O(n \log n)$, donde n es el numero de objetos a considerar.
- ~Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.

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 reconsidera la decisión ya tomada anteriormente respecto a la selección de un elemento a la vista de la decisión que se debe tomar en el instante actual.**

- c. Que la solución no sea la óptima.
13. Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?
- Programación dinámica.**
 - Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
 - El método voraz.
14. Sea la siguiente relación de recurrencia
- $$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$
- Si $T(n) \in O(n^2)$, ¿en cuál de estos tres casos nos podemos encontrar?
- $g(n) = n$
 - $g(n) = n^2$**
 - $g(n) = 1$
15. Un algoritmo recursivo basado en el esquema divide y vencerás...
- ... nunca tendrá una complejidad exponencial.
 - ... será más eficiente cuanto más equitativa sea la división en subproblemas.**
 - Las dos anteriores son ciertas.
16. En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...
- ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
 - Las otras dos opciones son ciertas.
 - ... 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.
- El problema de la mochila discreta.
 - El problema de las torres de Hanoi.**
 - 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...
- ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
 - ... una cota superior para el valor óptimo.
 - ... 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:
- El valor de la mochila continua correspondiente.
 - 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. ¿Qué debería ir en lugar de XXXXXXXX?

```
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. $f(n) \in O(v)$ 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 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. ¿Cuál 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...
- ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.
 - El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.
 - ... 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?
- Sí, ya que el mergesort es $O(n \log n)$ y este es $O(n)$.
 - Sólo su d log d > k n (donde k es una constante que depende de la implementación).**
 - 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?
- El problema de la asignación de tareas.
 - El problema del cambio.**
 - La mochila discreta sin restricciones adicionales.
39. Di cuál de estos tres algoritmos no es un algoritmo “divide y vencerás”
- Quicksort.
 - Mergesort.
 - El algoritmo de Prim.**
40. Si $f(n) \in O(n^3)$, ¿puede pasar que $f(n) \in O(n^2)$?
- No, porque n^3 no $\in O(n^2)$
 - Sólo para valores bajos de n
 - 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.  $n.\text{optimistic\_b}() \geq pb$
- \*b.  $n.\text{optimistic\_b}() \leq pb$**
- c.  $\text{pessimistic\_b}() \leq pb$

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:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT\left(\frac{n}{q}\right) + g(n) & \text{en otro caso} \end{cases}$$

(donde  $p$  y  $q$  son enteros mayores que 1). Di cuál de los siguientes esquemas algorítmicos produce de manera natural relaciones de recurrencia así.

- \*a. Divide y vencerás**
- b. Ramificación y poda.

c. Programación dinámica.

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. ¿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.  $n$ ,  $m[n]-1$ ,  $p$

49. Sea  $g(n) = \sum_{i=0}^K a_i n^i$  (hasta  $K$  con  $i = 0$ ). Di cuál de las siguientes afirmaciones es falsa:

- \*a. Las otras dos afirmaciones son ambas falsas.**
- b.  $g(n) \in \Theta(n^K)$
- c.  $g(n) \in \Omega(n^K)$

50. Si  $\lim_{n \rightarrow \infty} f(n)/g(n) = 0$  entonces...

- a. ...  $f(n) \in \Theta(g(n))$
- \*b. ...  $f(n) \in O(g(n))$**
- c. ...  $f(n) \in O(f(n))$

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 pierda el nodo que conduce a la solución óptima.

53. Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in O(n)$ , ¿en cuál de estos tres casos nos podemos encontrar?

- a.  $g(n) = n^2$
- b.  $g(n) = n$
- c.  $g(n) = 1$

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. ¿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.  $O(n \log n)$

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.  $\Theta(2^n)$
- b.  $\Omega(n)$  y  $O(n^2)$
- c.  $\Omega(n)$  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:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es falsa:

- a. Si  $g(n) \in O(n^2)$  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.

62. ¿Cuál es la definición correcta de  $O(f)$ ?

- a.  $O(f) = \{g : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n \geq n_0, f(n) \leq cg(n)\}$

- b.  $O(f) = \{g : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n \geq n_0, g(n) \leq cf(n)\}$
- c.  $O(f) = \{g : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n \geq n_0, f(n) \leq cg(n)\}$

63. Si  $f(n) \in O(n^2)$ , ¿podemos decir siempre que  $f(n) \in O(n^3)$ ?

- a. No, ya que  $n^2$  no es  $O(n^3)$
- b. Si, ya que  $n^2 \in O(n^3)$**
- 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 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.**

65. 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  $\Omega(n^2)$

66. La complejidad temporal de la solución de vuelta atrás al problema de la mochila discreta es...

- a. ... cuadrática en el caso peor.
- b. ... exponencial en cualquier caso.
- c. ... exponencial en el caso peor.**

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 del 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  $\lim_{n \rightarrow \infty} (f(n)/n^2) = k$ , y  $k \neq 0$ , ¿cuál de estas tres afirmaciones es falsa?

- a.  $f(n) \in O(n^3)$
- b.  $f(n) \in \Theta(n^3)$**
- c.  $f(n) \in \Theta(n^2)$

70. ¿Cuál es la definición correcta de  $\Omega(g)$ ?

- a.  $\Omega(g) = \{f : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n >= n_0, g(n) >= cf(n)\}$
- b.  $\Omega(g) = \{f : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n >= n_0, f(n) >= cg(n)\}$**
- c.  $\Omega(g) = \{f : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n >= n_0, f(n) >= cg(n)\}$

71. ¿Cuál es la definición correcta de  $\Omega(f)$ ?

- a.  $\Omega(f) = \{g : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n >= n_0, g(n) >= cf(n)\}$**
- b.  $\Omega(f) = \{g : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n >= n_0, f(n) >= cg(n)\}$
- c.  $\Omega(f) = \{g : N \rightarrow R^+ | \exists c \in R, \exists n_0 \in N, \forall n >= n_0, f(n) >= cg(n)\}$

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.  $n.optimistic\_b() \leq pb$
- b.  $n.pessimistic\_b() \leq pb$
- c.  $n.optimistic\_b() \geq pb$**

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.  $\Theta(\log_2(n)) = \Theta(\log_3(n))$
- c.  $\Theta(\log^2(n)) = \Theta(\log^3(n))$**

74. La función  $y$  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. **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.  $O(n^2) \subset O(2^{\log_2 n}) \subset O(2^n)$
- b.  $(4^{\log_2 n}) \subseteq O(n^2) \subset O(2^n)$
- c.  $O(2^{\log_2 n}) \subseteq O(n^2) \subset O(n!)$

82. Si  $f \in \Theta(g_1)$  y  $f \in \Theta(g_2)$  entonces

- a.  $f^2 \in \Theta(g_1 \cdot g_2)$
- b. **Las otras dos opciones son ambas ciertas**
- c.  $f \in \Theta(\max(g_1, g_2))$

83. La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es falsa:

- a. **Si  $g(n) \in \Theta(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.**
- b. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort.
- c. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación Mergesort.

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:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT\left(\frac{n}{a}\right) + g(n) & \text{en otro caso} \end{cases}$$

(donde  $p$  y  $a$  son enteros mayores que 1 y  $g(n) = n^k$ ), ¿qué tiene que ocurrir para que se cumpla  $T(n) \in \Theta(n^k \log_a n)$ ?

- a.  $p > a^k$
- b.  $p = a^k$
- c.  $p < a^k$

86. 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 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.  $\log(n^3) \neq \Theta(\log_3(n))$
- 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:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT\left(\frac{n}{a}\right) + g(n) & \text{en otro caso} \end{cases}$$

(donde  $p$  y  $a$  son enteros mayores que 1 y  $g(n) = n^k$ ), ¿qué tiene que ocurrir para que se cumpla  $T(n) \in \Theta(n^k)$ ?

- a.  $p > a^k$
- b.  $p < a^k$**
- c.  $p = a^k$

93. Si  $\lim_{n \rightarrow \infty} (f(n)/n^2) = k$ , y  $k \neq 0$ , ¿cuál de estas tres afirmaciones es cierta?

- a.  $f(n) \in \Omega(n^3)$
- b.  $f(n) \in \Theta(n^2)$**
- c.  $f(n) \in \Theta(n^3)$

94. Si  $f \in \Theta(g1)$  y  $f \in \Theta(g2)$  entonces

- a.  $f \in \Theta(g1 * g2)$
- b.  $f \notin \Theta(\max(g1, g2))$
- 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.  $\Theta(f) = O(f) \quad \square \quad \Omega(f)$
- b.  $\Omega(f) = \Theta(f) \quad \square \quad O(f)$
- c.  $O(f) = \Omega(f) \quad \square \quad \Theta(f)$

98. La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es cierta:

- a. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación quicksort.
- b. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso peor del algoritmo de ordenación quicksort.
- c. Si  $g(n) \in \Theta(1)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación mergesort.

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...
- ... debe recorrer siempre todo el árbol.
  - ... no se puede usar para resolver problemas de optimización
  - ... 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:
- El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
  - El valor de la mochila continua correspondiente.**
  - 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...
- $\Omega(n^2)$
  - $O(n)$**
  - $O(\log n)$
104. ¿En ramificación y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?
- Si, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista.
  - No, la cota optimista sólo se utiliza para determinar si una n-tupla es prometedora.
  - Sí, aunque no es una garantía de que sea una buena estrategia de búsqueda.**
105. ¿Qué estrategia de búsqueda es a priori más apropiada en un esquema de vuelta atrás?
- Explorar primero los nodos que están más completados.
  - En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.**
  - Explorar primero los nodos con mejor cota optimista.
106. Sea la relación de recurrencia
- $$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$
- Si  $T(n) \in O(n)$ , ¿en cuál de estos tres casos nos podemos encontrar?
- Las otras dos opciones son ambas ciertas.**
  - $g(n) = \log n$
  - $g(n) = \sqrt{n}$
107. Si  $f \in \Omega(g_1)$  y  $f \in \Omega(g_2)$  entonces
- $f \in \Omega(g_1 + g_2)$

- b.  $f \notin \Omega(\min(g_1, g_2))$
- c.  $f \in \Omega(g_1 * g_2)$

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  $\max(a, b) \in \Theta(1)$ )

```

float exa(vector<float>&v, vector<int>&p, int P, int i)
{
 float a, b;
 if (i>=0){
 if (p[i] <= P)
 a= v[i]+exa(v,p,P-p[i],i-1);
 else a= 0;
 b= exa(v,p,P,i-1);
 return max(a,b);
 }
 return 0;
}

```

Marcad la opción correcta.

- a. La complejidad temporal en el peor de los casos es  $O(n^2)$
- b. La complejidad temporal en el peor de los casos es  $O(2^n)$**
- c. La complejidad temporal en el mejor de los casos es  $\Omega(n^2)$

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 las 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  $T(n) \in \square(n^2)$**
- b.  $T(n) \in O(n)$  y  $T(n) \in \Theta(n)$
- c.  $T(n) \in \square(n)$  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:  $\text{pot}(x, n) = \text{pot}(x, n/2) * \text{pot}(x, n/2)$ ; ¿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  $\Theta(n \times m)$  y espacial  $\Theta(n \times m)$
- b. Temporal  $\Theta(n \times m)$  y espacial  $\Theta(\min\{n, m\})$**
- c. Temporal  $\Theta(\max\{n, m\})$  y espacial  $\Theta(\max\{n, m\})$

125. ¿Qué se deduce de  $f(n)$  y  $g(n)$  si se cumple  $\lim_{n \rightarrow \infty} (f(n)/g(n)) = k$ , con  $k \neq 0$ ?

- a.  $f(n) \in O(g(n))$  y  $g(n) \in O(f(n))$**
- b.  $f(n) \in O(g(n))$  pero  $g(n) \notin O(f(n))$
- c.  $g(n) \in O(f(n))$  pero  $f(n) \notin O(g(n))$

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  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás, ¿cuál sería la forma más adecuada de representar las posibles soluciones ?

- a. Una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.
- b. Un vector de booleanos.**
- c. Un par de enteros que indiquen los cortes realizados y el valor acumulado.

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.  $\Theta(n) \subset \Theta(n^2)$
- b.  $n + n \log n \in \Omega(n)$
- c.  $O(2^{\log n}) \subset O(n^2)$

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. ¿Cuál es el orden de complejidad que se obtuvo?

- a.  $n \log^2 n$
- b.  $n^2$
- c.  $n \log n$

136. Se desea ordenar una lista enlazada de  $n$  elementos haciendo uso del algoritmo mergesort. En este caso, al tratarse de una lista, la complejidad temporal asintótica de realizar la división en subproblemas resulta ser lineal con el tamaño de esa lista. ¿Cuál sería entonces el coste temporal de realizar dicha ordenación?

- a.  $\Theta(n \log n)$
- b.  $\Theta(n^2)$
- c. **Ninguna de las otras dos opciones es cierta.**

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  $\Theta(1)$
- 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  $\Theta(n^2)$ .
  - b. La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en  $\Theta(n^2)$ .
  - 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.

Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para tratar de acelerar la búsqueda calculamos, para cada uno de los nodos visitados, una cota pesimista y otra optimista. ¿Cómo debería comportarse el algoritmo en el caso en el que el valor de ambas coincidan?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. El algoritmo debería terminar pues ya ha encontrado la solución. X
- c. Aún así es necesario expandir ese nodo.
- d. Ese nodo no debería expandirse.

La respuesta correcta es: Ese nodo no debería expandirse.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?  $f(n) = \begin{cases} 1 & n = 1 \\ n + 2f(n - 1) & n > 1 \end{cases}$

Seleccione una:

- a. No contesto (equivalente a no marcar nada). ✗
- b.  $f(n) \in O(2^n)$
- c.  $f(n) \in \Omega(2^n)$
- d.  $f(n) \in \Theta(2^n)$

La respuesta correcta es:  $f(n) \in \Omega(2^n)$

Con respecto al tamaño del problema, ¿Cuál es la complejidad temporal de la siguiente función?

```
unsigned f(unsigned n) {
 if(n < 2) return n;
 unsigned sum=0;
 for(int i = 0; i < 10; i++)
 sum+=n;
 return sum;
}
```

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $\Theta(1)$  ✓
- c. Ninguna de las otras dos opciones es correcta.
- d.  $\Theta(n)$

La respuesta correcta es:  $\Theta(1)$

¿Cuál de estas afirmaciones es falsa?

Seleccione una:

- 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. Hay problemas de optimización para los cuales se puede obtener siempre la solución óptima utilizando una estrategia voraz.
- c. Todos los problemas de optimización tienen una solución voraz que es óptima sea cual sea la instancia a resolver. ✓
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: Todos los problemas de optimización tienen una solución voraz que es óptima sea cual sea la instancia a resolver.

## La solución de programación dinámica iterativa del problema de la mochila discreta ..

Seleccione una:

- a. ... tiene la restricción de que los pesos tienen que ser enteros positivos. ✓
- b. No contesto (equivalente a no marcar nada).
- c. ... tiene un coste temporal asintótico exponencial.
- d. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.

La respuesta correcta es: ... tiene la restricción de que los pesos tienen que ser enteros positivos.

Por qué muchos algoritmos voraces presentan complejidades temporales en  $O(n \log n)$ ?

Seleccione una:

- a. Porque el proceso desección de los elementos que se incorporarán a la solución es siempre  $O(n \log n)$ .
- 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 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 es estrictamente inferior a  $O(n \log n)$ . ✗
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: 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  $O(n \log n)$ .

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Utilizando la técnica programación dinámica iterativa se pretende conocer la dificultad del camino más favorable. ¿Cuál es la mejor complejidad espacial y temporal que se puede conseguir?

Seleccione una:

- a. Espacial  $\Theta(n)$  y temporal  $\Theta(n \cdot m)$ .
- b. No contesto (equivalente a no marcar nada).
- c.  $\Theta(n + m)$ , tanto espacial como temporal.
- d. Espacial  $\Theta(\min(n, m))$  y temporal  $\Theta(n \cdot m)$ .



La respuesta correcta es: Espacial  $\Theta(\min(n, m))$  y temporal  $\Theta(n \cdot m)$ .

Sea el vector  $v[8] = \{8, 6, 4, 5, 4, 5, 2, 2\}$ . Indica cuál de las siguientes opciones es cierta. (se asume la notación del lenguaje C/C++ en la que el primer elemento del vector está en la posición 0, es decir, en  $v[0]$ ).

Seleccione una:

- a. El vector  $v$  no es un montículo máximo porque el elemento  $v[3]=5$  debe ser "flotado" (desplazado hacia la izquierda).
- b. El vector  $v$  no es un montículo máximo porque el elemento  $v[2]=4$  debe ser "hundido" (desplazado hacia la derecha).
- c. No contesto (equivalente a no marcar nada).
- d. El vector  $v$  es un montículo máximo.

La respuesta correcta es: El vector  $v$  no es un montículo máximo porque el elemento  $v[2]=4$  debe ser "hundido" (desplazado hacia la derecha).

Uno de estos tres problemas no tiene una solución trivial y eficiente que siga el esquema voraz.

Seleccione una:

- a. El problema del cambio sin restricciones en cuanto al número de monedas disponibles de cada clase. ✓
- b. El problema de la mochila continua.
- c. No contesto (equivalente a no marcar nada).
- d. El problema de la mochila discreta sin limitación en la carga máxima de la mochila.

La respuesta correcta es: El problema del cambio sin restricciones en cuanto al número de monedas disponibles de cada clase.

Se la recurrencia:  $T(n) = \begin{cases} 1 & n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & n > 1 \end{cases}$  donde donde  $g(n)$  es una función polinómica. De las siguientes afirmaciones, o bien dos son ciertas y una es falsa, o bien dos son falsas y una es cierta. Marque la opción que en este sentido es diferente a las demás.

Seleccione una:

- a. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso peor del algoritmo de ordenación Quicksort.
- b. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort.
- c. No contesto (equivalente a no marcar nada).
- d. Si  $g(n) \in \Theta(1)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Mergesort.

La respuesta correcta es: Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort.

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n){
 if (n<=1)
 return 0;
 unsigned sum = desperdicio (n/2) + desperdicio (n/2) + desperdicio (n/2);
 for (unsigned i=1; i<n-1; i++)
 for (unsigned j=1; j<=i; j++)
 for (unsigned k=1; k<=j; k++)
 sum+=i*j*k;
 return sum;
}
```

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $\Theta(n^3)$
- c.  $\Theta(3^n)$
- d.  $\Theta(n^3 \log n)$



La respuesta correcta es:  $\Theta(n^3)$

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Utilizando la técnica memoización se pretende conocer la dificultad del camino más favorable. ¿Cuál es la mejor complejidad espacial y temporal que se puede conseguir?

Seleccione una:

- a. Espacial  $\Theta(n)$  y temporal  $\Theta(2^{\max(n,m)})$ .
- b. No contesto (equivalente a no marcar nada).
- c.  $\Theta(n \cdot m)$ , tanto espacial como temporal.
- d. Espacial  $\Theta(n)$  y temporal  $\Theta(n \cdot m)$ .



La respuesta correcta es:  $\Theta(n \cdot m)$ , tanto espacial como temporal.

¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?

Seleccione una:

- a. Que el algoritmo sería más lento pues se explorarían más nodos de los necesarios.
- b. No contesto (equivalente a no marcar nada).
- c. Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.
- d. Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.



La respuesta correcta es: Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. una estrategia voraz puede ser la única alternativa.
- c. es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione.
- d. podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.

La respuesta correcta es: una estrategia voraz puede ser la única alternativa.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para tratar de acelerar la búsqueda calculamos una cota para cada uno de los nodos visitados que consiste en resolver el mismo problema pero aplicando la técnica programación dinámica con la condición de que el camino debe pasar por la casilla asociada al nodo que acabamos de visitar. ¿Qué podemos decir de esa cota?

Seleccione una:

- a. Que en realidad no se trata de una cota, ni pesimista ni optimista.
- b. Que es una cota pesimista pero no optimista. ✗
- c. No contesto (equivalente a no marcar nada).
- d. Que es una cota pesimista y optimista al mismo tiempo.

La respuesta correcta es: Que en realidad no se trata de una cota, ni pesimista ni optimista.

En un mapa similar al del problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur se pretende conocer el número de caminos distintos que hay desde la casilla  $(1, 1)$  hasta una casilla cualquiera  $(i, j)$  que pertenece al mapa. El mapa tiene  $n$  filas y  $m$  columnas ¿Cuál sería la recurrencia matemática más apropiada para abordarlo mediante Divide y vencerás?

Seleccione una:

- a.  $\text{nc}(i, j) = \begin{cases} 0 & \text{si } i < 1 \text{ o } j < 1 \\ 1 & \text{si } i = 1 \text{ y } j = 1 \\ \text{nc}(i - 1, j - 1) + \text{nc}(i, j - 1) + \text{nc}(i - 1, j) & \text{en cualquier otro caso} \end{cases}$
- b.  $\text{nc}(n, m) = \begin{cases} 0 & \text{si } n < i \text{ o } m < j \\ 1 & \text{si } i = 1 \text{ y } j = 1 \\ \text{nc}(n - 1, m - 1) + \text{nc}(n, m - 1) + \text{nc}(n - 1, m) & \text{en cualquier otro caso} \end{cases}$
- c. No contesto (equivalente a no marcar nada).
- d.  $\text{nc}(i, j) = \begin{cases} 0 & \text{si } n < i \text{ o } m < j \\ 1 & \text{si } i = 1 \text{ y } j = 1 \\ \text{nc}(i - 1, j - 1) + \text{nc}(i, j - 1) + \text{nc}(i - 1, j) + 1 & \text{en cualquier otro caso} \end{cases}$

La respuesta correcta es:

$$\text{nc}(i, j) = \begin{cases} 0 & \text{si } i < 1 \text{ o } j < 1 \\ 1 & \text{si } i = 1 \text{ y } j = 1 \\ \text{nc}(i - 1, j - 1) + \text{nc}(i, j - 1) + \text{nc}(i - 1, j) & \text{en cualquier otro caso} \end{cases}$$

Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para tratar de acelerar la búsqueda utilizamos un algoritmo de programación dinámica aplicado al mismo problema pero restringiendo los movimientos a tres: Este, Sureste y Sur. Con este algoritmo calculamos dos valores: 1: La dificultad del mejor camino desde una casilla cualquiera ( $i,j$ ) hasta el destino; 2: La dificultad del mejor camino desde el origen hasta esa casilla ( $i,j$ ). ¿con respecto al nodo que representa esa casilla, qué se puede decir de estos dos valores?

Seleccione una:

- a. Con el segundo se puede componer una cota pesimista; con el primero no se puede componer una cota optimista.
- b. Con el primero se puede componer una cota pesimista; con el segundo no se puede componer una cota optimista.
- c. El primero es una cota pesimista; el segundo es una cota optimista.
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: Con el primero se puede componer una cota pesimista; con el segundo no se puede componer una cota optimista.

Sea el problema "Camino de coste mínimo" con 8 movimientos: las ocho casilla colindantes. Utilizando la técnica backtracking se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para el nodo inicial calculamos una cota pesimista y otra optimista ¿Cómo debería comportarse el algoritmo en el caso de que el valor de ambas coincidan?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. No debería continuar.
- c. Eso no es posible: alguna de las cotas estaría mal calculada.
- d. Debería continuar tratando de mejorar el valor obtenido.



La respuesta correcta es: No debería continuar.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Utilizando un algoritmo voraz se pretende conocer la dificultad del camino más favorable y para ello se va seleccionando una casilla escogida al azar de entre las tres posibles, hasta llegar al destino. ¿Qué podemos decir de este algoritmo?

Seleccione una:

- a. Que aún siendo voraz existen otros criterios de selección con mejores resultados a priori.
- b. Que es voraz puesto que las decisiones no se replantean.
- c. Que no es voraz.
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es: Que no es voraz.

En función del parámetro  $n$  ¿cuál es la complejidad temporal de la siguiente función?

```
int f(int n){
 int k=0;
 for (int i = 1; i < n; i*=3)
 for (int j=i; j > 0; j-=3)
 k++;
 return k;
}
```

Seleccione una:

- a.  $\Theta(n^2)$
- b.  $\Theta(n)$
- c.  $\Theta(n \log(n))$
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es:  $\Theta(n)$

Si  $\lim_{n \rightarrow \infty} (g(n)/f(n))$  resulta ser una constante positiva no nula, cuál de las siguientes expresiones NO puede darse?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $g(n) \notin \Theta(f(n))$
- c.  $f(n) \in \Theta(g(n))$
- d.  $f(n) \in \Omega(g(n))$  y  $g(n) \in \Omega(f(n))$



La respuesta correcta es:  $g(n) \notin \Theta(f(n))$

La versión de Quicksort que utiliza como pivote el elemento del vector que ocupa la posición central ...

Seleccione una:

- a. .... se comporta mejor cuando el vector está ordenado al contrario de como se quiere ordenar.
- b. No contesto (equivalente a no marcar nada). X
- c. .... se comporta peor cuando el vector ya está ordenado.
- d. .... no presenta caso mejor y peor para instancias del mismo tamaño.

La respuesta correcta es: .... se comporta mejor cuando el vector está ordenado al contrario de como se quiere ordenar.

Seleccione una:

- a. Mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante.
- b. No contesto (equivalente a no marcar nada). ✗
- c. La complejidad espacial de todos ellos es lineal con el tamaño del vector a ordenar.
- d. Mergesort y Heapsort tienen complejidad espacial lineal con el tamaño del vector a ordenar, la de Quicksort es constante.

la respuesta correcta es: Mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante.

Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. A igualdad de coste temporal, ¿qué se prefiere una buena cota pesimista o una buena cota optimista?

Seleccione una:

- a. No contesto (equivalente a no marcar nada). ✗
- b. No hay distinción, la más rentable siempre será la que mejor se aproxime a la solución del nodo.
- c. La optimista, una buena cota optimista siempre será más rentable que una buena cota pesimista.
- d. La pesimista, una buena cota pesimista siempre será más rentable que una buena cota optimista.

La respuesta correcta es: No hay distinción, la más rentable siempre será la que mejor se aproxime a la solución del nodo.

Sea el problema "Camino de coste mínimo" con tres movimientos Este, Sureste y Sur. Se pretende encontrar el camino más favorable con la menor complejidad temporal posible. ¿Cuál sería la más ajustada para realizar todo el proceso?

Seleccione una:

- a.  $O(n \cdot m + n)$
- b.  $O(n \cdot m + m)$
- c. No contesto (equivalente a no marcar nada).
- d.  $O(n \cdot m + n + m)$

La respuesta correcta es:  $O(n \cdot m + n + m)$

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Se pretende listar todos los caminos posibles entre dos casillas cualesquiera. ¿Qué técnica algorítmica deberíamos utilizar?

Seleccione una:

- a. Backtracking.
- b. Programación dinámica.
- c. Ramificación y poda.
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: Backtracking.

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.

Seleccione una:

a.  $\Theta(\log^2(n)) = \Theta(\log^3(n))$



b. No contesto (equivalente a no marcar nada).

c.  $\Theta(\log(n^2)) = \Theta(\log(n^3))$

d.  $\Theta(\log_2(n)) = \Theta(\log_3(n))$

La respuesta correcta es:  $\Theta(\log^2(n)) = \Theta(\log^3(n))$

¿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?

Seleccione una:

- a. La combinación de las soluciones a los subproblemas.
- b. El número de llamadas recursivas que se hacen.
- c. No contesto (equivalente a no marcar nada).
- d. La división del problema en subproblemas.



La respuesta correcta es: La división del problema en subproblemas.

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de la mochila continua correspondiente.
- b. El valor de una mochila que contiene todos los objetos restantes aunque se pase del peso máximo permitido.
- c. No contesto (equivalente a no marcar nada).
- d. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.



La respuesta correcta es: El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. En cuanto a la complejidad espacial y temporal, ¿Cuál de los siguientes esquemas algorítmicos resulta más eficiente para resolverlo?

Seleccione una:

- a. Ramificación y poda
- b. Divide y vencerás
- c. Programación dinámica
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es: Programación dinámica

Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para tratar de acelerar la búsqueda calculamos, para cada uno de los nodos visitados, una cota que consiste en suponer que la dificultad del camino restante es la suma de las dificultades de todas las casillas del mapa ¿De qué tipo de cota se trata?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. No es cota, ni pesimista ni optimista.
- c. Optimista.
- d. Pesimista.



La respuesta correcta es: No es cota, ni pesimista ni optimista.

Se quiere desarrollar un programa que compruebe si es posible que un caballo de ajedrez, mediante una secuencia de sus movimientos permitidos, recorra todas las casillas de un tablero  $N \times N$  a partir de una determinada casilla dada como entrada y sin repetir ninguna casilla. De entre las estrategias que se citan, ¿cuál sería la eficiente para resolver el problema?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Vuelta atrás.
- c. Programación dinámica.
- d. Algoritmo voraz.



La respuesta correcta es: Vuelta atrás.

En un algoritmo de ramificación y poda, si la lista de nodos vivos no está clasificada de forma apropiada ...

Seleccione una:

- a. ... podría ocurrir que se exploren nodos de forma innecesaria.
- b. No contesto (equivalente a no marcar nada).
- c. ... podría ocurrir que se pade el nodo que conduce a la solución óptima.
- d. ... podría ocurrir que se descarten nodos factibles.



La respuesta correcta es: ... podría ocurrir que se exploren nodos de forma innecesaria.

En cuanto a la complejidad temporal asintótica de estas dos funciones, ¿cuál de las siguientes afirmaciones es correcta?

```
unsigned long pot2_1(unsigned n){
 if (n==0)
 return 1;
 return 2 * pot2_1(n-1);
}
```

```
unsigned long pot2_2(unsigned n){
 if (n==0)
 return 1;
 if (n%2==0)
 return pot2_2(n/2)*pot2_2(n/2);
 else
 return 2*pot2_2(n/2)*pot2_2(n/2);
}
```

Seleccione una:

- a. La segunda es más eficiente que la primera. X
- b. La primera es más eficiente que la segunda.
- c. Ambas son equivalentes en cuanto a eficiencia
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: Ambas son equivalentes en cuanto a eficiencia

Seleccione una:

- a.  $f(n) = 2n \cdot \log_2 n - \sum_{i=0}^{\log_2 n} i \cdot 2^i \in O(n)$
- b. Ninguna de las otras dos formulaciones es correcta.
- c. No contesto (equivalente a no marcar nada). \*
- d.  $f(n) = 2n - \sum_{i=0}^{\log_2 n} i \cdot 2^i \in O(n)$

La respuesta correcta es:  $f(n) = 2n \cdot \log_2 n - \sum_{i=0}^{\log_2 n} i \cdot 2^i \in O(n)$

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur pero modificado de manera que el punto de partida es una casilla cualquiera  $(i, j)$  del mapa con nombre  $M$  ( $1 \leq i \leq n$  y  $1 \leq j \leq m$ ) y el destino la casilla  $(n, m)$ . Utilizando la técnica divide y vencerás se pretende conocer la dificultad del camino más favorable. ¿Cuál sería la recurrencia matemática más apropiada?

Seleccione una:

- a.  $mcp(i, j) = \begin{cases} \infty & \text{si } n < i \text{ o } m < j \\ M[i][j] & \text{si } n = i \text{ y } m = j \\ \min(mcp(i - 1, j - 1), mcp(i, j - 1), mcp(i - 1, j)) + M[i][j] & \text{en cualquier otro caso} \end{cases}$
- b.  $mcp(n, m) = \begin{cases} 0 & \text{si } n < i \text{ o } m < j \\ M[i][j] & \text{si } n = i \text{ y } m = j \\ \min(mcp(n - 1, m - 1), mcp(n, m - 1), mcp(n - 1, m)) & \text{en cualquier otro caso} \end{cases}$
- c.  $mcp(n, m) = \begin{cases} \infty & \text{si } n < i \text{ o } m < j \\ M[n][m] & \text{si } n = i \text{ y } m = j \\ \min(mcp(n - 1, m - 1), mcp(n, m - 1), mcp(n - 1, m)) + M[n][m] & \text{en cualquier otro caso} \end{cases}$
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es:

$$mcp(n, m) = \begin{cases} \infty & \text{si } n < i \text{ o } m < j \\ M[n][m] & \text{si } n = i \text{ y } m = j \\ \min(mcp(n - 1, m - 1), mcp(n, m - 1), mcp(n - 1, m)) + M[n][m] & \text{en cualquier otro caso} \end{cases}$$

Sea  $g(n) = \sum_{i=0}^k a_i n^i$ . ¿Cuál de las siguientes afirmaciones es falsa?

Seleccione una:

- a.  $g(n) \in \Omega(n^k)$
- b.  $g(n) \in \Theta(n^k)$
- c.  $g(n) \in \Theta(n^{k+1})$
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es:  $g(n) \in \Theta(n^k)$

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

Seleccione una:

- a. una estrategia voraz puede ser la única alternativa.
- b. No contesto (equivalente a no marcar nada).
- c. podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.
- d. es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione.



La respuesta correcta es: una estrategia voraz puede ser la única alternativa.

Dado un vector de tamaño  $n$  de números enteros. ¿Con qué complejidad temporal se puede determinar si sus elementos están dispuestos formando un montículo de mínimos?

Seleccione una:

- a.  $O(n)$
- b.  $O(\log n)$
- c. No contesto (equivalente a no marcar nada).
- d.  $O(1)$



La respuesta correcta es:  $O(n)$

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para tratar de acelerar la búsqueda calculamos una cota para cada uno de los nodos visitados que consiste en resolver el mismo problema pero aplicando la técnica programación dinámica con la condición de que el camino debe pasar por la casilla asociada al nodo que acabamos de visitar. ¿Qué podemos decir de esa cota?

Seleccione una:

- a. Que en realidad no se trata de una cota, ni pesimista ni optimista.
- b. Que es una cota pesimista pero no optimista.
- c. Que es una cota pesimista y optimista al mismo tiempo.
- d. No contesto (equivalente a no marcar nada). X

La respuesta correcta es: Que es una cota pesimista y optimista al mismo tiempo.

Sea  $g(n) = \sum_{j=0}^{\log n} n\left(\frac{1}{2}\right)^j$ . Suponiendo  $n \rightarrow \infty$ , ¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $g(n) \in \Theta(n \log n)$
- b. No contesto (equivalente a no marcar nada). ×
- c.  $g(n) \in \Omega(n \log n)$
- d.  $g(n) \in O(n)$

La respuesta correcta es:  $g(n) \in O(n)$

Qué diferencia (entre otras) hay entre el algoritmo de Prim y el de Kruskal?

Seleccione una:

- a. El algoritmo de Prim es voraz y el de Kruskal no.
- b. No contesto (equivalente a no marcar nada).
- c. Aún siendo el grafo de partida totalmente conexo, el algoritmo de Kruskal garantiza la solución óptima mientras que el de Prim sólo garantiza un subóptimo.
- d. El subgrafo que paso a paso va generando el algoritmo de Prim siempre contiene una única componente conexa mientras que el de Kruskal no tiene por qué. ✓

La respuesta correcta es: El subgrafo que paso a paso va generando el algoritmo de Prim siempre contiene una única componente conexa mientras que el de Kruskal no tiene por qué.

Sea el problema "Camino de coste mínimo" con 8 movimientos: las ocho casilla colindantes. Utilizando la técnica backtracking se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para el nodo inicial calculamos una cota pesimista y otra optimista ¿Cómo debería comportarse el algoritmo en el caso de que el valor de ambas coincidan?

Seleccione una:

- a. Eso no es posible: alguna de las cotas estaría mal calculada.
- b. No contesto (equivalente a no marcar nada).
- c. No debería continuar.
- d. Debería continuar tratando de mejorar el valor obtenido. X

La respuesta correcta es: No debería continuar.

Sea la recurrencia:  $T(n) = \begin{cases} 1 & n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & n > 1 \end{cases}$  donde  $g(n)$  es una función polinómica. 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 diferente a las demás.

Seleccione una:

- a. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort. 
- b. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso peor del algoritmo de ordenación Quicksort.
- c. No contesto (equivalente a no marcar nada).
- d. Si  $g(n) \in \Theta(1)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Mergesort.

La respuesta correcta es: Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort.

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n){
if (n<=1)
 return 0;
unsigned sum = desperdicio (n/2) + desperdicio (n/2);
for (unsigned i=1; i<n-1; i++)
 for (unsigned j=1; j<=i; j++)
 for (unsigned k=1; k<=j; k++)
 sum+=i*j*k;
return sum;
}
```

Seleccione una:

- a.  $\Theta(2^n)$
- b. No contesto (equivalente a no marcar nada).
- c.  $\Theta(n^3 \log n)$  ✗
- d.  $\Theta(n^3)$  ←

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur pero modificado de manera que el punto de partida es una casilla cualquiera  $(i, j)$  del mapa con nombre  $M$  ( $1 \leq i \leq n$  y  $1 \leq j \leq m$ ) y el destino la casilla  $(n, m)$ . Utilizando la técnica divide y vencerás se pretende conocer la dificultad del camino más favorable. ¿Cuál sería la recurrencia matemática más apropiada?

Seleccione una:

a. No contesto (equivalente a no marcar nada).

b.

$$\text{mcp}(n, m) = \begin{cases} \infty & \text{si } n < i \text{ o } m < j \\ M[n][m] & \text{si } n = i \text{ y } m = j \\ \min(\text{mcp}(n - 1, m - 1), \text{mcp}(n, m - 1), \text{mcp}(n - 1, m)) + M[n][m] & \text{en cualquier otro caso} \end{cases}$$

c.

$$\text{mcp}(i, j) = \begin{cases} \infty & \text{si } n < i \text{ o } m < j \\ M[i][j] & \text{si } n = i \text{ y } m = j \\ \min(\text{mcp}(i - 1, j - 1), \text{mcp}(i, j - 1), \text{mcp}(i - 1, j)) + M[i][j] & \text{en cualquier otro caso} \end{cases}$$

d.

$$\text{mcp}(n, m) = \begin{cases} 0 & \text{si } n < i \text{ o } m < j \\ M[i][j] & \text{si } n = i \text{ y } m = j \\ \min(\text{mcp}(n - 1, m - 1), \text{mcp}(n, m - 1), \text{mcp}(n - 1, m)) & \text{en cualquier otro caso} \end{cases}$$

La respuesta correcta es:  $\text{mcp}(n, m) = \begin{cases} \infty & \text{si } n < i \text{ o } m < j \\ M[n][m] & \text{si } n = i \text{ y } m = j \\ \min(\text{mcp}(n - 1, m - 1), \text{mcp}(n, m - 1), \text{mcp}(n - 1, m)) + M[n][m] & \text{en cualquier otro caso} \end{cases}$

En cuanto a la complejidad temporal asintótica de estas dos funciones, ¿cuál de las siguientes afirmaciones es correcta?

```
unsigned long pot2_1(unsigned n){
```

```
 if (n==0)
 return 1;
 return 2 * pot2_1(n-1);
}
```

```
unsigned long pot2_2(unsigned n){
```

```
 if (n==0)
 return 1;
 if (n%2==0)
 return pot2_2(n/2)*pot2_2(n/2);
 else
 return 2*pot2_2(n/2)*pot2_2(n/2);
}
```

Seleccione una:

- a. La segunda es más eficiente que la primera.
- b. No contesto (equivalente a no marcar nada).
- c. Ambas son equivalentes en cuanto a eficiencia
- d. La primera es más eficiente que la segunda.



La respuesta correcta es: Ambas son equivalentes en cuanto a eficiencia

El coste temporal de un algoritmo se ajusta a la siguiente ecuación de recurrencia:  $T(0) = 1$ ;  $T(n) = n + \sum_{j=0}^{n-1} T(j)$  si  $n \geq 1$ . ¿qué coste temporal asintótico (o complejidad temporal) tendrá el algoritmo?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $O(n!)$
- c.  $O(n^2)$
- d.  $O(2^n)$



La respuesta correcta es:  $O(2^n)$

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Se pretende encontrar el camino más favorable con la menor complejidad temporal posible. ¿Cuál sería la más ajustada para realizar todo el proceso?

Seleccione una:

- a.  $O(n \cdot m + n + m)$
- b.  $O(n \cdot m + m)$
- c. No contesto (equivalente a no marcar nada).
- d.  $O(n \cdot m + n)$



La respuesta correcta es:  $O(n \cdot m + n + m)$

Si  $\lim_{n \rightarrow \infty} (g(n)/f(n))$  resulta ser cero, cuál de las siguientes expresiones puede darse?

Seleccione una:

- a. Ninguna de las otras dos opciones son ciertas. ✓
- b.  $f(n) \in \Omega(g(n))$  y  $g(n) \in \Omega(f(n))$
- c.  $f(n) \in \Theta(g(n))$
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: Ninguna de las otras dos opciones son ciertas.

Tenemos una función recursiva con la siguiente cabecera:

```
double f(const double &)
```

Con sólo esta información, ¿cuál podría ser una definición adecuada para el almacén?

Seleccione una:

- a. `vector<int> A;`
- b. `vector<double> A;`
- c. No contesto (equivalente a no marcar nada).
- d. Ninguna de las otras dos opciones son válidas.



La respuesta correcta es: Ninguna de las otras dos opciones son válidas.

Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. A igualdad de coste temporal, ¿qué se prefiere una buena cota pesimista o una buena cota optimista?

Seleccione una:

- a. La optimista, una buena cota optimista siempre será más rentable que una buena cota pesimista.
- b. La pesimista, una buena cota pesimista siempre será más rentable que una buena cota optimista.
- c. No hay distinción, la más rentable siempre será la que mejor se aproxime a la solución del nodo.
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es: No hay distinción, la más rentable siempre será la que mejor se aproxime a la solución del nodo.

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de la mochila continua correspondiente.
- b. El valor de una mochila que contiene todos los objetos restantes aunque se pase del peso máximo permitido.
- c. No contesto (equivalente a no marcar nada).
- d. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.



La respuesta correcta es: El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

La versión de Quicksort que utiliza como pivote el elemento del vector que ocupa la primera posición ...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. No contesto (equivalente a no marcar nada).
- c. ... se comporta peor cuando el vector ya está ordenado.
- d. ... no presenta caso mejor y peor para instancias del mismo tamaño.



La respuesta correcta es: ... se comporta peor cuando el vector ya está ordenado.

Sea el problema "Camino de coste mínimo" con ocho movimientos (todas las casillas colindantes) pero modificado de manera que ahora se pretende encontrar el camino de máxima dificultad desde el origen hasta la casilla destino. ¿Qué cambios habría que hacer con respecto al problema sin modificar resuelto mediante ramificación y poda?

Seleccione una:

- a. Se podría intercambiar la cota pesimista por la optimista y viceversa.
- b. No contesto (equivalente a no marcar nada).
- c. La cota optimista se podría reutilizar sin hacerle ningún cambio, pero utilizándola como cota pesimista. X
- d. Se podría reutilizar la cota pesimista, pero adaptándola al nuevo problema.

La respuesta correcta es: Se podría reutilizar la cota pesimista, pero adaptándola al nuevo problema.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Utilizando la técnica programación dinámica iterativa se pretende conocer la dificultad del camino más favorable. ¿Cuál es la mejor complejidad espacial y temporal que se puede conseguir?

Seleccione una:

- a. Espacial  $\Theta(\min(n, m))$  y temporal  $\Theta(n \cdot m)$ .
- b. No contesto (equivalente a no marcar nada).
- c.  $\Theta(n + m)$ , tanto espacial como temporal.
- d. Espacial  $\Theta(n)$  y temporal  $\Theta(n \cdot m)$ .



La respuesta correcta es: Espacial  $\Theta(\min(n, m))$  y temporal  $\Theta(n \cdot m)$ .

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... tiene un coste temporal asintótico exponencial.
- c. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.
- d. ... tiene la restricción de que los pesos tienen que ser enteros positivos.



La respuesta correcta es: ... tiene la restricción de que los pesos tienen que ser enteros positivos.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Se pretende listar todos los caminos posibles entre dos casillas cualesquiera ¿Qué técnica algorítmica deberíamos utilizar?

Seleccione una:

- a. Programación dinámica.
- b. Ramificación y poda.
- c. Backtracking.
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es: Backtracking.

Con respecto a  $n$ , ¿Cuál es la complejidad temporal del siguiente fragmento de código?

```
int a = 0;
for(int i = 0; i < n; i += 2)
 for(int j = 0; j < 10; j += 2)
 a += A[i][j];
}
```

Seleccione una:

- a.  $\Theta(n)$
- b.  $\Theta(n^2)$
- c. No contesto (equivalente a no marcar nada).
- d.  $\Theta(\log n)$



La respuesta correcta es:  $\Theta(n)$

Se quiere desarrollar un programa que compruebe si es posible que un caballo de ajedrez, mediante una secuencia de sus movimientos permitidos, recorra todas las casillas de un tablero  $N \times N$  a partir de una determinada casilla dada como entrada y sin repetir ninguna casilla. De entre las estrategias que se citan, ¿cuál sería la eficiente para resolver el problema?

Seleccione una:

- a. Algoritmo voraz.
- b. No contesto (equivalente a no marcar nada).
- c. Programación dinámica.
- d. Vuelta atrás.

La respuesta correcta es: Vuelta atrás.

Con respecto a la complejidad espacial de los algoritmos de ordenación Quicksort, Heapsort y Mergesort:

Seleccione una:

- a. Mergesort y Heapsort tienen complejidad espacial lineal con el tamaño del vector a ordenar, la de Quicksort es constante.
- b. Las complejidad espacial de todos ellos es lineal con el tamaño del vector a ordenar.
- c. Mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante.
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es: Mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Utilizando un algoritmo voraz se pretende conocer la dificultad del camino más favorable y para ello se va seleccionando una casilla escogida al azar de entre las tres posibles, hasta llegar al destino. ¿Qué podemos decir de este algoritmo?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Que no es voraz.
- c. Que es voraz puesto que las decisiones no se replantean.
- d. Que aún siendo voraz existen otros criterios de selección con mejores resultados a priori.

✗

La respuesta correcta es: Que no es voraz.

En función del parámetro  $n$  ¿cuál es la complejidad temporal de la siguiente función?

```
int f(int n){
 int k=0;
 for (int i = 1; i < n; i*=3)
 for (int j=i; j > 0; j-=3)
 k++;
 return k;
}
```

Seleccione una:

a.  $\Theta(n \log(n))$

b.  $\Theta(n^2)$

c. No contesto (equivalente a no marcar nada).

d.  $\Theta(n)$



Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. En cuanto a la complejidad espacial y temporal, ¿Cuál de los siguientes esquemas algorítmicos resulta más eficiente para resolverlo?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Programación dinámica
- c. Ramificación y poda
- d. Divide y vencerás



La respuesta correcta es: Programación dinámica

¿Cuál de las siguientes expresiones se corresponde mejor con el cálculo de la complejidad temporal asintótica en el caso peor del algoritmo que construye un Heap mínimo a partir de un vector (array) de números?

Seleccione una:

- a.  $f(n) = 2n \cdot \log_2 n - \sum_{i=0}^{\log_2 n} i \cdot 2^i \in O(n)$
- b. Ninguna de las otras dos formulaciones es correcta.
- c.  $f(n) = 2n - \sum_{i=0}^{\log_2 n} i \cdot 2^i \in O(n)$
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es:  $f(n) = 2n \cdot \log_2 n - \sum_{i=0}^{\log_2 n} i \cdot 2^i \in O(n)$

Se pretende resolver el problema del viajante de comercio (travelling salesman problem) mediante Ramificación y poda. ¿Cuál de las siguientes acciones resulta ser mejor cota optimista para aplicarla a los nodos intermedios?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Asumir que ya no quedan más vértices por visitar y cerrar el camino desde el último vértice visitado.
- c. Obtener el árbol de recubrimiento de mínimo coste a los vértices aún no visitados.
- d. Completar el recorrido pendiente mediante un algoritmo voraz.



La respuesta correcta es: Obtener el árbol de recubrimiento de mínimo coste a los vértices aún no visitados.

¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?

Seleccione una:

- a. Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.
- b. No contesto (equivalente a no marcar nada).
- c. Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.
- d. Que el algoritmo sería más lento pues se explorarían más nodos de los necesarios.



La respuesta correcta es: Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.

En un algoritmo de ramificación y poda, si la lista de nodos vivos no está clasificada de forma apropiada ...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... podría ocurrir que se descarten nodos factibles.
- c. ... podría ocurrir que se exploren nodos de forma innecesaria.
- d. ... podría ocurrir que se pade el nodo que conduce a la solución óptima.



La respuesta correcta es: ... podría ocurrir que se exploren nodos de forma innecesaria.

¿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?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. La combinación de las soluciones a los subproblemas.
- c. La división del problema en subproblemas.
- d. El número de llamadas recursivas que se hacen.



La respuesta correcta es: La división del problema en subproblemas.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Utilizando la técnica memoización se pretende conocer la dificultad del camino más favorable. ¿Cuál es la mejor complejidad espacial y temporal que se puede conseguir?

Seleccione una:

- a. Espacial  $\Theta(n)$  y temporal  $\Theta(n \cdot m)$ .
- b.  $\Theta(n \cdot m)$ , tanto espacial como temporal.
- c. Espacial  $\Theta(n)$  y temporal  $\Theta(2^{\max(n,m)})$ .
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es:  $\Theta(n \cdot m)$ , tanto espacial como temporal.

Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para tratar de acelerar la búsqueda calculamos, para cada uno de los nodos visitados, una cota pesimista y otra optimista. ¿Cómo debería comportarse el algoritmo en el caso en el que el valor de ambas coincidan?

Seleccione una:

- a. Aún así es necesario expandir ese nodo.
- b. El algoritmo debería terminar pues ya ha encontrado la solución.
- c. Ese nodo no debería expandirse.
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es: Ese nodo no debería expandirse.

Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para tratar de acelerar la búsqueda utilizamos un algoritmo de programación dinámica aplicado al mismo problema pero restringiendo los movimientos a tres: Este, Sureste y Sur. Con este algoritmo calculamos dos valores: 1: La dificultad del mejor camino desde una casilla cualquiera  $(i,j)$  hasta el destino; 2: La dificultad del mejor camino desde el origen hasta esa casilla  $(i,j)$ . ¿con respecto al nodo que representa esa casilla, qué se puede decir de estos dos valores?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Con el primero se puede componer una cota pesimista; con el segundo no se puede componer una cota optimista. ✓
- c. El primero es una cota pesimista; el segundo es una cota optimista.
- d. Con el segundo se puede componer una cota pesimista; con el primero no se puede componer una cota optimista.

La respuesta correcta es: Con el primero se puede componer una cota pesimista; con el segundo no se puede componer una cota optimista.

¿Para cuál de estos problemas de optimización existe una solución voraz?

Seleccione una:

- a. El problema de la mochila discreta.
- b. El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
- c. No contesto (equivalente a no marcar nada).
- d. 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.



La respuesta correcta es: El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.

Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para tratar de acelerar la búsqueda calculamos, para cada uno de los nodos visitados, una cota que consiste en suponer que la dificultad del camino restante es la suma de las dificultades de todas las casillas del mapa ¿De qué tipo de cota se trata?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Pesimista.
- c. No es cota, ni pesimista ni optimista.
- d. Optimista.



La respuesta correcta es: Optimista.

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.

Seleccione una:

a.  $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$

b. No contesto (equivalente a no marcar nada).

c.  $(4^{\log_2(n)}) \subset O(n) \subset O(2^n)$

d.  $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$



La respuesta correcta es:  $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$

Por qué muchos algoritmos voraces presentan complejidades temporales en  $O(n \log n)$ ?

Seleccione una:

- a. Porque el proceso de selección de los elementos que se incorporarán a la solución es siempre  $O(n \log n)$ .
- 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 es estrictamente inferior a  $O(n \log n)$ .
- c. 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  $O(n \log n)$ . ✓
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: 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  $O(n \log n)$ .

En un mapa similar al del problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur se pretende conocer el número de caminos distintos que hay desde la casilla  $(1, 1)$  hasta una casilla cualquiera  $(i, j)$  que pertenece al mapa. El mapa tiene  $n$  filas y  $m$  columnas ¿Cuál sería la recurrencia matemática más apropiada para abordarlo mediante Divide y vencerás?

Seleccione una:

a. No contesto (equivalente a no marcar nada).

b.  $\text{nc}(i, j) = \begin{cases} 0 & \text{si } n < i \text{ o } m < j \\ 1 & \text{si } i = 1 \text{ y } j = 1 \\ \text{nc}(i - 1, j - 1) + \text{nc}(i, j - 1) + \text{nc}(i - 1, j) + 1 & \text{en cualquier otro caso} \end{cases}$

c.  $\text{nc}(n, m) = \begin{cases} 0 & \text{si } n < i \text{ o } m < j \\ 1 & \text{si } i = 1 \text{ y } j = 1 \\ \text{nc}(n - 1, m - 1) + \text{nc}(n, m - 1) + \text{nc}(n - 1, m) & \text{en cualquier otro caso} \end{cases}$

d.  $\text{nc}(i, j) = \begin{cases} 0 & \text{si } i < 1 \text{ o } j < 1 \\ 1 & \text{si } i = 1 \text{ y } j = 1 \\ \text{nc}(i - 1, j - 1) + \text{nc}(i, j - 1) + \text{nc}(i - 1, j) & \text{en cualquier otro caso} \end{cases}$



La respuesta correcta es:  $\text{nc}(i, j) = \begin{cases} 0 & \text{si } i < 1 \text{ o } j < 1 \\ 1 & \text{si } i = 1 \text{ y } j = 1 \\ \text{nc}(i - 1, j - 1) + \text{nc}(i, j - 1) + \text{nc}(i - 1, j) & \text{en cualquier otro caso} \end{cases}$

¿Qué esquema algorítmico utiliza el algoritmo de ordenación Quicksort?

\*Divide y Vencerás

Programación Dinámica

Backtracking.

Ante un problema que presenta una solución recursiva siempre podemos aplicar:

\*Divide y vencerás

Programación Dinámica

Cualquiera de las dos anteriores.

En cual de los siguientes casos no se puede aplicar el esquema Divide y Vencerás:

Cuando los subproblemas son de tamaños muy diferentes

Cuando el problema no cumple el principio de optimalidad

\*Se puede aplicar en ambos casos.

Dado el algoritmo de búsqueda binaria, supongamos que, en vez de dividir la lista de elementos en dos mitades del mismo tamaño, la dividamos en dos partes de tamaños 1/3 y 2/3. El coste de este algoritmo:

Es el mismo que el original

\*Es mayor que el del original

Es menor que el del original.

Si  $n$  es el número de elementos del vector, el coste del algoritmo Mergesort es:

$O(n^2)$  y  $\Omega(n \log(n))$

\* $\Theta(n \log(n))$

$\Theta(n^2)$ .

Un problema se puede resolver por Divide y Vencerás siempre que:

\*Cumpla el principio de optimalidad

Cumpla el teorema de reducción

Ninguna de las anteriores.

La serie de números de Fibonacci se define de la siguiente forma:

$$fib(n) = \begin{cases} 1 & n \leq 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$$

Para implementar esta función podemos emplear...

Divide y vencerás

Programación dinámica

\*Cualquiera de las dos anteriores.

La serie de números de Fibonacci se define de la siguiente forma:

$$fib(n) = \begin{cases} 1 & n \leq 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$$

¿Qué implementación de entre las siguientes supone el menor coste?

Divide y vencerás

\*Programación dinámica

Ambas tienen el mismo coste asintótico.

El problema de la mochila, ¿Puede solucionarse de forma óptima empleando la estrategia de divide y vencerás?

Sólo para el caso de la mochila con fraccionamiento

\*Sólo para el caso de la mochila sin fraccionamiento

Sí, se puede aplicar para ambos casos.

Para que un problema de optimización se pueda resolver mediante programación dinámica es necesario que:

\*Cumpla el principio de optimalidad

Cumpla el teorema de reducción

Cumpla las dos anteriores.

Dada una solución recursiva a un problema, ¿Cómo podemos evitar la resolución de los mismos subproblemas muchas veces?

Resolver los subproblemas de mayor a menor y guardar su resultado en una tabla, inicializándola con los problemas pequeños

\*Resolver los subproblemas de menor a mayor y guardar su resultado en una tabla, inicializándola con los problemas pequeños

Resolver los subproblemas de mayor a menor y guardar su resultado en una tabla, inicializándola con los problemas más grandes.

Si aplicamos programación dinámica a un problema que también tiene solución por divide y vencerás podemos asegurar que...

El coste temporal se reduce y el espacial aumenta con respecto a la solución por DyV

El coste temporal aumenta y el espacial se reduce con respecto a la solución por DyV

\*Ninguna de las anteriores.

¿Cuándo utilizaremos Programación Dinámica en lugar de Divide y Vencerás?

Cuando se incrementa la eficacia

\*Cuando se incrementa la eficiencia

Cuando se reduce el coste espacial.

En programación dinámica, dónde almacenaremos los valores de los problemas resueltos?

En un vector unidimensional

En un vector bidimensional

\*Depende del problema.

Supongamos el problema de la mochila resuelto mediante Programación Dinámica y particularizado para n elementos y un peso máximo trasportable de P. ¿Es necesario calcular valores para toda la matriz auxiliar para obtener el resultado?

Si

\*No

Depende de los valores de n y P.

Un problema de optimización cuya solución se puede expresar mediante una secuencia de decisiones cumple el principio de optimalidad si, dada una secuencia óptima:

Existe una subsecuencia de esa solución que corresponde a la solución óptima de su subproblema asociado

Existe al menos una subsecuencia de esa solución que corresponde a la solución óptima de su subproblema asociado

\*Cualquier subsecuencia de esa solución corresponde a la solución óptima de su subproblema asociado.

La programación dinámica, para resolver un problema, aplica la estrategia...

\*Se resuelven los problemas más pequeños y, combinando las soluciones, se obtienen las soluciones de problemas sucesivamente más grandes hasta llegar al problema original

Se descompone el problema a resolver en subproblemas más pequeños, que se resuelven independientemente para finalmente combinar las soluciones de los subproblemas para obtener la solución del problema original

Ninguna de las anteriores.

¿Qué esquema de programación es el adecuado para resolver el problema k-ésimo mínimo en un vector?

Programación Dinámica

\*Divide y Vencerás

Ninguno de los dos.

Si  $n$  es el número de elementos de un vector. La solución de menor coste al problema de encontrar su k-ésimo mínimo tiene la siguiente complejidad:

$\Omega(n)$  y  $O(n \log(n))$

\* $\Omega(n)$  y  $O(n^2)$

Ninguna de las dos.

Si  $n$  es el número de elementos de un vector. Podemos encontrar una solución al problema de encontrar su k-ésimo que esté acotada superiormente por:

\* $O(n^3)$

$O(n)$

Ninguna de las anteriores.

Dada la solución recursiva al problema de encontrar el k-ésimo mínimo de un vector.  
Cada llamada recursiva, ¿cuántas nuevas llamadas recursivas genera?

\*una o ninguna

dos o ninguna

una o dos.

La solución al problema de encontrar el k-ésimo mínimo de un vector pone en práctica la siguiente estrategia:

Ordenar totalmente el vector

\*Ordena parcialmente el vector

No ordena ningún elemento del vector.

¿Qué esquema de programación es el adecuado para resolver el problema de la búsqueda binaria?

Programación Dinámica

\*Divide y Vencerás

Ninguno de los dos.

Si  $n$  es el número de elementos de un vector. La solución de menor coste al problema de la búsqueda binaria tiene la siguiente complejidad:

$\Omega(\log(n))$  y  $O(n \log(n))$

$\Theta(n \log(n))$

\* $\Omega(1)$  y  $O(\log(n))$ .

¿Con qué esquema de programación obtenemos algoritmos que calculan la distancia de edición entre dos cadenas?

\*Programación Dinámica

Divide y Vencerás

Ambos.

Disponemos de dos cadenas de longitudes  $m$  y  $n$ . Si resolvemos el problema de la

distancia de edición mediante programación dinámica, ¿De qué tamaño debemos definir la matriz que necesitaremos?

(m-1) x (n-1)

m x n

\* (m+1) x (n+1).

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

El árbol de cobertura de coste mínimo de un grafo conexo

\*El problema de la mochila discreta o sin fraccionamiento

El problema de la mochila continua o con fraccionamiento.

Los algoritmos de programación dinámica hacen uso...

... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.

\*... de que se puede ahorrar cálculos guardando resultados anteriores en un almacén.

... de una estrategia trivial consistente en examinar todas las soluciones posibles.

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles

\*El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores

El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.

Supongamos que un informático quiere subir a una montaña y como no es una persona normal 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?

\*Un algoritmo voraz

Un algoritmo divide y vencerás

Un algoritmo de programación dinámica.

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que...

... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.

\*... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos.

El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

\*Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar.

Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.

Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log(n))$ , donde  $n$  es el número de objetos a considerar.

Dada la suma de la recurrencia:

$$T(n) = \begin{cases} 1 & n=0 \\ \sum_{k=0}^{n-1} T(k) & n>0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

$$T(n) \in \Theta(n^2)$$

$$T(n) \in \Theta(n!)$$

$$*T(n) \in \Theta(2^n).$$

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

El método voraz

Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño

\*Programación dinámica.

En el método voraz...

... siempre se encuentra solución pero puede que no sea óptima

\*... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar

... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

¿Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores?

Ya no se garantizaría la solución óptima pero sí una factible

Habrá que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible

\*La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor estructura para el almacén?

int A

int A[]

\*int A [] [].

Se pretende implementar mediante programación dinámica iterativa la función recursiva: ¿Cuál es la mejor estructura para el almacén?

```

unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}

```

\*int A[]

int A

int A[] [].

¿Cuál de estas estrategias para calcular el n-ésimo elemento de la serie de Fibonacci es más eficiente?

$$(f(n) = f(n-1) + f(n-2), f(1) = f(2) = 1)$$

La estrategia voraz

\*Programación Dinámica

Las dos estrategias citadas serían similares en cuanto a eficiencia .

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante un proceso recursivo. Este problema es candidato a tener una solución alternativa basada en...

Un algoritmo voraz

\*Un algoritmo de programación dinámica

Un algoritmo de estilo Divide y Vencerás.

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible

El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles

\*El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.

La solución de programación dinámica iterativa del problema de la mochila discreta...

... calcula menos veces al valor de la mochila que la correspondiente solución de programación dinámica recursiva

... tiene un coste temporal asintótico exponencial con respecto al número de objetos

\*... tiene la restricción de que los valores tienen que ser enteros positivos.

El problema de encontrar un árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado...

sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo

no se puede resolver en general con una estrategia voraz

\*se puede resolver siempre con una estrategia voraz.

Si ante un problema de decisión existe un criterio de selección voraz entonces...

al menos una solución factible está garantizada

la solución óptima está garantizada

\*ninguna de las otras dos opciones es cierta.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A,2+B);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

O(1)

O(y^2)

\* $O(y)$ .

El valor que se obtiene con el método voraz para el problema de la mochila discreta es...

\*una cota inferior para el valor óptimo que a veces puede ser igual a este  
una cota inferior para el valor óptimo, pero que nunca coincide con este  
una cota superior para el valor óptimo.

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Meter primero los elementos de menor peso

\*Meter primero los elementos de mayor valor específico o valor por unidad de peso

Meter primero los elementos de mayor valor.

La eficiencia de los algoritmos voraces se basa en el hecho de que...

antes de tomar una decisión se comprueba si satisface las restricciones del problema con antelación, las posibles decisiones se ordenan de mejor a peor

\*las decisiones tomadas nunca se reconsideran.

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que...

en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos

el número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica

\*en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos.

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc... Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es FALSA.

Hace una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$

Es posible evitar hacer la evaluación exhaustiva "de fuerza bruta" guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente

\*Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ .

Se pretende implementar mediante programación dinámica iterativa la función recursiva: ¿Cuál es la mejor complejidad espacial que se puede conseguir?

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

O( $x^2$ )

\*O(1)

O(x).

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

O( $y^2$ )

O(1)

\*O(y).

La programación dinámica...

En algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos

\*Las otras dos opciones son ciertas

Normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores.

Dado un problema de optimización, el método voraz...

siempre obtiene una solución factible

siempre obtiene la solución óptima

\*garantiza la solución óptima sólo para determinados problemas.

Un algoritmo recursivo basado en el esquema divide y vencerás...

Las demás opciones son verdaderas

\*... será más eficiente cuanto más equitativa sea la división en subproblemas.

... nunca tendrá una complejidad exponencial.

Sea  $f(n)$  la solución de la relación de recurrencia  $f(n) = f(n/2) + 1$ ;  $f(1) = 1$ . Indicad cuál de estas tres expresiones es cierta:

$$f(n) \in \Theta(n)$$

$$f(n) \in \Theta(n \log(n))$$

$$*f(n) \in \Theta(\log(n)).$$

De las siguientes expresiones, o bien dos son verdaderas y una falsa, o bien dos son falsas y una verdadera. Marca la que (en este sentido) es distinta a las otras dos.

$$O(n^2) \subset O(2^{(\log_2(n))}) \subset O(2^n)$$

$$O(n^2) \subset O(2^{(\log_2(n))}) \subseteq O(2^n)$$

$$*O(2^{(\log_2(n))}) \subset O(n^2) \subset O(2^n).$$

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```

unsigned desperdicio
(unsigned n){
 if (n<=1)
 return 0;
 unsigned sum = desperdicio
 (n/2) + desperdicio (n/2);
 for (unsigned i=1; i<n-1;
 i++)
 for (unsigned j=1;
 j<=i; j++)
 sum+=i*j;
 return sum;
}

```

$\Theta(2^n)$

$\Theta(n^2 \log(n))$

\* $\Theta(n^2)$ .

De las siguientes expresiones, o bien dos son verdaderas y una falsa, o bien dos son falsas y una verdadera. Marca la que (en este sentido) es distinta a las otras dos.

$\Omega(n^3) \subset \Omega(n^2)$

\* $\Theta(n^2) \subset \Theta(n^3)$

$O(n^2) \subset O(n^3)$ .

Para la complejidad de un algoritmo presente caso mejor y peor distintos...

\*... es condición necesaria que existan instancias distintas del problema con el mismo tamaño.

... es condición suficiente que existan instancias distintas del problema con el mismo tamaño.

... es condición necesaria y suficiente que existan instancias distintas del problema con el mismo tamaño.

Se dispone de un vector de tamaño  $n$  cuyos elementos están de antemano organizados

formando un montículo (heap). ¿Cuál sería la complejidad temporal de reorganizar el vector para que quede ordenado?

O( n )

O( log(n) )

\*O( n log(n) ).

Indica cuál es la complejidad, en función de n, del fragmento siguiente:

```
k=n/4;
for(int i = k; i < n - k; i++){
 A[i] = 0;
 for(int j = i - k; j < i + k; j++)
 A[i] += B[j];
}
```

O( n log(n) )

O( n )

\*O( n^2 ).

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n = 1 \\ n^2 + 3f(n/3) & n > 1 \end{cases}$$

f(n) ∈ Θ( n )

f(n) ∈ Θ( n^2 log(n) )

\*f(n) ∈ Θ( n^2 ).

La versión Quicksort que utiliza como pivote el elemento del vector que ocupa la primer posición...

... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo

... se comporta mejor cuando el vector está ya ordenado

\*... se comporta peor cuando el vector ya está ordenado.

Dado el siguiente fragmento de código...

```
for(i=0;i<n;i++) for(j=1;j< i;j+=2) s+=i*j;
```

Indica cuál de las siguientes expresiones es la que mejor refleja su complejidad temporal.

$\sum \log(j)$

\*Ninguna de las otras dos opciones son correctas

$\sum \log(i)$ .

Un vector de enteros de tamaño n tiene sus elementos estructurados en forma de montículo (heap). ¿Cuál es la complejidad temporal en el mejor de los casos de borrar el primer elemento del vector y reorganizarlo posteriormente para que siga manteniendo la estructura de montículo?

Ninguna de las otras dos opciones es correcta

\*Constante con el tamaño del vector

$O(n)$ .

Un problema de tamaño n puede transformarse en tiempo  $O(n)$  en siete de tamaño  $n/7$ ; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante. ¿Cuál de estas clases de coste temporal asintótico es la más ajustada?

\* $O(n \log(n))$

$O(n^2)$

$O(n)$ .

Indica cuál es la complejidad en función de n del fragmento siguiente:

```
int a = 0;
for(int i = 0; i < n; i++)
 for(int j = i; j > 0; j /=2)
 a += A[i][j];
```

$O(n^2)$

$O(n)$

\* $O(n \log(n))$ .

Un programa con dos bucles anidados uno dentro del otro. El primero hace n iteraciones aproximadamente y el segundo la mitad, tarda un tiempo...

\* $O(n^2)$

$O(n \log(n))$

$O(n\sqrt{n})$ .

Sea  $f(n)$  la solución de la relación de recurrencia  $f(n) = 2*f(n/2) + n$ ;  $f(1) = 1$ . Indicad cuál de estas tres expresiones es cierta:

$f(n) \in \Theta(n^2)$

\* $f(n) \in \Theta(n \log(n))$

$f(n) \in \Theta(n)$ .

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n = 1 \\ n + 3f(n/3) & n > 1 \end{cases}$$

$f(n) \in \Theta(n^3)$

$f(n) \in \Theta(n)$

\* $f(n) \in \Theta(n \log(n))$ .

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n){
 if (n<=1)
 return 0;
 unsigned sum = desperdicio (n/2) + desperdicio (n/2);
 for (unsigned i=1; i<n-1; i++)
 for (unsigned j=1; j<=i; j++)
 for (unsigned k=1; k<=j; k++)
 sum+=i*j*k;
 return sum;
}
```

$\Theta(2^n)$

\* $\Theta(n^3)$

$\Theta(n^3 \log(n))$ .

Di cuál de estos resultados de coste temporal asintótico es falsa:

\*La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en  $\Omega(n^2)$

La ordenación de un vector usando el algoritmo Quicksort requiere en el peor caso un tiempo en  $\Omega(n^2)$

La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en  $O(\log(n))$ .

Sea  $f(n)$  la solución de la relación de recurrencia...

$$f(n) = 2f(n - 1) + 1; f(1) = 1.$$

Indicad cuál de estas tres expresiones es cierta:

\* $f(n) \in \Theta(2^n)$

$f(n) \in \Theta(n^2)$

$f(n) \in \Theta(n)$ .

De las siguientes expresiones, o bien dos son verdaderas y una falsa, o bien dos son falsas y una verdadera. Marca la que (en este sentido) es distinta a las otras dos.

$\Omega(n^2) \subset \Omega(n^3)$

$\Theta(n^2) \subset \Theta(n^3)$

\* $O(n^2) \subset O(n^3)$ .

De las siguientes expresiones, o bien dos son verdaderas y una falsa, o bien dos son falsas y una verdadera. Marca la que (en este sentido) es distinta a las otras dos.

$\Theta(\log^2(n)) = \Theta(\log(n^2))$

$\Theta(\log^2(n)) = \Theta(\log(n))$

\* $\Theta(\log(n)) = \Theta(\log(n^2))$ .

Un problema de tamaño  $n$  puede transformarse en tiempo  $O(n^3)$  en ocho de tamaño

$n/2$ ; 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?

$O(n^2 \log(n))$

$O(n^3)$

\* $O(n^3 \log(n))$ .

La eficiencia de los algoritmos voraces se basa en el hecho de que...

\*las decisiones tomadas nunca se reconsideran

antes de tomar una decisión se comprueba si satisface las restricciones del problema con antelación, las posibles decisiones se ordenan de mejor a peor.

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

\*Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar

Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log(n))$ , donde  $n$  es el número de objetos a considerar

Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles

\*El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores

El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

El árbol de cobertura de coste mínimo de un grafo conexo

\*El problema de la mochila discreta o sin fraccionamiento

El problema de la mochila continua o con fraccionamiento.

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

\*El fontanero diligente y la mochila continua

El fontanero diligente y el problema del cambio

El fontanero diligente y la asignación de tareas.

La solución óptima al problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado...

\*Puede construirlo tanto vértice a vértice como arista a arista

Debe construirlo vértice a vértice: arista a arista no puede ser

Debe construirlo arista a arista: vértice a vértice no puede ser.

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que...

En la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos

\*En la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos

El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.

¿Qué mecanismo se usa para acelerar el algoritmo de Prim?

El TAD "Union-find"

Mantener una lista de los arcos ordenados según su peso

\*Mantener para cada vértice su "padre" más cercano.

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x > y) return 1;
 return x*(y-1) + f(x,y-2);
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

O(y - x), tanto temporal como espacial

Temporal O(x\*y) y espacial O(x)

\*Ninguna de las otras dos opciones es correcta.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if(x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 1; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

O( $x^2$ )

\*O(x)

O(1).

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad temporal que se puede conseguir?

O(y)

\*O(x\*y)

O(x).

¿Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores?

Ya no se garantizaría la solución óptima pero sí una factible

Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible

\*La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna.

En el método voraz...

\*Es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.

Siempre se encuentra solución pero puede que no sea la óptima.

El dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

Los algoritmos de programación dinámica hacen uso ...

... de una estrategia trivial consistente en examinar todas las soluciones posibles.

... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.

\*... de que se puede ahorrar cálculos guardando resultados anteriores en un almacén.

Cuando se calculan los coeficientes binomiales usando la recursión

Cuando se calculan los coeficientes binomiales usando la recursión  $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$ , con  $\binom{n}{0} = \binom{n}{n} = 1$ , qué problema se da y cómo se puede resolver?

La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.

\*Se repiten muchos cálculos y ello se puede evitar usando programación dinámica.

Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.

El valor que se obtiene con el método voraz para el problema de la mochila discreta es...

Una cota inferior para el valor óptimo, pero que nunca coincide con este.

Una cota superior para el valor óptimo.

\*Una cota inferior para el valor óptimo que a veces puede ser igual a este.

El problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado...

\*se puede resolver siempre con una estrategia voraz

no se puede resolver en general con una estrategia voraz

sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo.

¿Por qué se utiliza el TAD "Union-find" en el algoritmo de Kruskal?

\*Para comprobar si un arco forma ciclos

Para comprobar si un vértice ya ha sido visitado

Para comprobar si dos vértices son equivalentes.

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Es posible evitar hacer la evaluación exhaustiva "de fuerza bruta" guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.

\*Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$

Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .

¿Cuál de estas estrategias voraces obtiene siempre un mejor valor para la mochila discreta?

meter primero los elementos de mayor valor específico o valor por unidad de peso

meter primero los elementos de mayor valor

\*ninguna de las otras dos opciones es cierta.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

$O(y^2)$

\* $O(y)$

$O(x^2)$ .

La programación dinámica...

Normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores

En algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos

\*Las otras dos opciones son ciertas.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A,2+B);
}
```

¿Cuál es la mejor complejidad temporal que se puede conseguir?

O(y)

O(x)

\*O(x\*y).

Se pretende implementar mediante programación dinámica iterativa la función

recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;

 unsigned m = 0;

 for (unsigned y = 1; y < x; y++)
 m = max(m, v[y] + f(x-y, v));

 return m;
}
```

¿Cuál es la mejor complejidad temporal que se puede conseguir?

\*O( $x^2$ )

O(x)

O( $x \cdot y$ ).

¿Cuál de estas estrategias para calcular el  $n$ -ésimo elemento de la serie Fibonacci ( $f(n) = f(n-1) + f(n-2)$ ,  $f(1) = f(2) = 1$ ) es más eficiente?

La estrategia voraz

\*Programación dinámica

Las dos estrategias citadas serían similares en cuanto a eficiencia.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

$O(y^2)$

$O(x^2)$

\* $O(y)$ .

El método voraz se emplea en la resolución de problemas de selección y optimización en los que se pretende encontrar:

\*Una solución que satisfaga unas restricciones y optimice una cierta función objetivo.

Todas las soluciones que satisfagan unas restricciones.

La dos respuestas anteriores son correctas.

Voraz siempre da solución óptima:

Al problema de la mochila sin fraccionamiento.

\*Al problema de la mochila con fraccionamiento.

A los dos.

En el método Voraz, aunque las decisiones son irreversibles, podemos asegurar que:

Siempre obtendremos la solución óptima.

Siempre obtendremos una solución factible.

\*Sólo obtendremos la solución óptima para algunos problemas.

Dado un problema de optimización y un algoritmo Voraz que lo soluciona, ¿cuándo podemos estar seguros de que la solución obtenida será óptima?:

\*Cuando demostremos formalmente que el criterio conduce a una solución óptima para cualquier instancia del problema.

Voraz siempre encuentra solución óptima.

En ambos casos. Las dos son correctas.

Si aplicamos un algoritmo voraz que no nos garantiza la solución óptima sobre un problema entonces...

Obtendremos una solución factible.

\*Puede que no encuentre ninguna solución aunque ésta exista.

Si el problema tiene solución óptima, el esquema voraz nos garantiza que la encuentra.

El problema de la mochila, ¿encuentra su solución óptima empleando la estrategia voraz?:

\*Sólo para el caso de la mochila con fraccionamiento

Sólo para el caso de la mochila sin fraccionamiento

En cualquiera de los casos anteriores.

Dado un grafo G que representa las poblaciones de la provincia de Alicante de más de 20.000 habitantes junto con todas las carreteras de conexión entre ellas. Queremos obtener el recorrido que nos permita pasar por todas estas ciudades una única vez y volver al punto de origen recorriendo el mínimo número de kilómetros. Si aplicamos una estrategia voraz sobre este grafo obtendremos...

\*Una solución factible

La solución óptima

Puede que no encuentre ninguna solución aunque ésta exista.

Al aplicar backtracking obtenemos la solución óptima a un problema:

Siempre

En algunos casos

\*Sólo cuando el problema cumple el principio de Optimalidad.

Si aplicamos un esquema backtracking que no nos garantiza la solución óptima sobre un problema entonces

Obtendremos una solución factible.

Puede que no encuentre ninguna solución aunque ésta exista.

\*Ninguna de las anteriores.

Backtracking es aplicable a problemas de selección y optimización en los que:

El espacio de soluciones es un conjunto infinito.

\*El espacio de soluciones es un conjunto finito.

En cualquiera de los casos.

En un problema resuelto por backtracking, el conjunto de valores que pueden tomar las componentes de la tupla solución, ha de ser:

Infinito.

\*finito

continuo.

Al aplicar vuelta atrás a la solución de problemas, obtenemos algoritmos con costes computacionales:

Polinómicos.

\*Exponenciales.

Los dos son correctos. Depende del problema.

Vuelta atrás se emplea en la resolución de problemas de optimización en los que se pretende encontrar:

Una solución que satisfaga unas restricciones y optimice una cierta función objetivo.

Todas las soluciones que satisfagan unas restricciones.

\*La dos respuestas anteriores son correctas.

Backtracking procede a obtener la solución a un problema de optimización empleando la siguiente estrategia:

Genera todas las combinaciones de la solución y selecciona la que optimiza la función objetivo.

\*Genera todas las soluciones factibles y selecciona la que optimiza la función objetivo.

Genera una solución factible empleando un criterio óptimo.

Backtracking es una técnica de resolución general de problemas basada en:

\*La búsqueda sistemática de soluciones.

La construcción directa de la solución.

Ninguna de las anteriores.

Backtracking genera las soluciones posibles al problema:

\*Mediante el recorrido en profundidad del árbol que representa el espacio de soluciones.

Mediante el recorrido en anchura del árbol que representa el espacio de soluciones

Ninguna de las anteriores.

El problema de la mochila, ¿puede solucionarse empleando vuelta atrás?:

Sólo para el caso de la mochila con fraccionamiento

\*Sólo para el caso de la mochila sin fraccionamiento

Se puede aplicar para ambos casos.

El problema del viajante de comercio puede resolverse correctamente empleando estos esquemas de programación:

\*Solo backtracking

Empleando cualquiera de estos: Voraz y backtracking.

Sólo programación dinámica.

El tiempo de ejecución de un algoritmo de ramificación y poda depende de:

La instancia del problema

La función de selección de nodos para su expansión

\*De ambos.

Dado un problema resuelto mediante backtracking y mediante ramificación y poda, el coste computacional de la solución por ramificación y poda, en comparación con la de backtracking es:

\*Igual

Mayor

Menor.

¿Cuál de estas afirmaciones es falsa?

\*Backtracking inspecciona todo el espacio de soluciones de un problema mientras que Ramificación y poda no.

La complejidad en el peor caso de las soluciones Backtracking y ramificación y poda a un mismo problema es la misma.

Para un mismo problema, ramificación y poda explora siempre un número de nodos menor o igual que backtracking.

El problema de la asignación de turnos tiene solución óptima voraz aplicando la siguiente estrategia:

**EL PROBLEMA DE LA ASIGNACION DE TURNOS.**

Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asigne una prioridad. El número de alumnos es N. Se dispone una matriz cuadrada M con N filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre -1 y N-1) indicando dicha prioridad (un valor -1 indica que no quiere o no puede estar con la persona de la columna correspondiente, 0 indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona). Ningún alumno puede formar grupo consigo mismo.

Se pretende encontrar una solución para satisfacer el número máximo de alumnos según su orden de preferencia. Suponiendo que la matriz M ya existe, diseñar un algoritmo que resuelva el problema de forma óptima

Seleccionamos los alumnos en orden descendente de preferencia respetando las restricciones de cabida de cada turno.

Seleccionamos los alumnos en orden ascendente de preferencia respetando las restricciones de cabida de cada turno

\*El problema no tiene solución óptima voraz.

El problema de la asignación de turnos tiene solución óptima empleando:

**EL PROBLEMA DE LA ASIGNACION DE TURNOS.**

Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asigne una prioridad. El número de alumnos es N. Se dispone una matriz cuadrada M con N filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre -1 y N-1) indicando dicha prioridad (un valor -1 indica que no quiere o no puede estar con la persona de la columna correspondiente, 0 indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona). Ningún alumno puede formar grupo consigo mismo.

Se pretende encontrar una solución para satisfacer el número máximo de alumnos según su orden de preferencia. Suponiendo que la matriz M ya existe, diseñar un algoritmo que resuelva el problema de forma óptima

\*Backtracking

Voraz

Ambos.

El problema de la asignación de turnos tiene solución...

**EL PROBLEMA DE LA ASIGNACION DE TURNOS.**

Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asignen una prioridad. El número de alumnos es N. Se dispone una matriz cuadrada M con N filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre -1 y N-1) indicando dicha prioridad (un valor -1 indica que no quiere o no puede estar con la persona de la columna correspondiente, 0 indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona) . Ningún alumno puede formar grupo consigo mismo.

Se pretende encontrar una solución para satisfacer el número máximo de alumnos según su orden de preferencia. Suponiendo que la matriz M ya existe, diseñar un algoritmo que resuelva el problema de forma óptima

Optima mediante backtracking

Aproximada (sub-óptima) mediante voraz

\*Ambas.

Dada la solución recursiva mediante vuelta atrás al problema de la asignación de turnos ¿cuántas nuevas llamadas recursivas genera cada llamada recursiva?

**EL PROBLEMA DE LA ASIGNACION DE TURNOS.**

Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asignen una prioridad. El número de alumnos es N. Se dispone una matriz cuadrada M con N filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre -1 y N-1) indicando dicha prioridad (un valor -1 indica que no quiere o no puede estar con la persona de la columna correspondiente, 0 indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona) . Ningún alumno puede formar grupo consigo mismo.

Se pretende encontrar una solución para satisfacer el número máximo de alumnos según su orden de preferencia. Suponiendo que la matriz M ya existe, diseñar un algoritmo que resuelva el problema de forma óptima

una o dos

una o ninguna

\*ninguna de las anteriores.

El problema de la asignación de turnos resuelto mediante backtracking tiene una complejidad:

**EL PROBLEMA DE LA ASIGNACION DE TURNOS.**

Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asignen una prioridad. El número de alumnos es N. Se dispone una matriz cuadrada M con N filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre -1 y N-1) indicando dicha prioridad (un valor -1 indica que no quiere o no puede estar con la persona de la columna correspondiente, 0 indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona) . Ningún alumno puede formar grupo consigo mismo.

Se pretende encontrar una solución para satisfacer el número máximo de alumnos según su orden de preferencia. Suponiendo que la matriz M ya existe, diseñar un algoritmo que resuelva el problema de forma óptima

\*Exponencial

Polinómica

Ninguna de la dos.

Cuando se resuelve 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 en el caso peor es la mejor que nos podemos encontrar?

\* $O(2^n)$

$O(n!)$

$O(n^2)$ .

Al resolver el problema del viajante de comercio mediante vuelta atrás y asumiendo un grafo de n vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Se multiplica n por la distancia de la arista más corta que nos queda por considerar

Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las n aristas más cortas

\*Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.

Se desea obtener todas las permutaciones de una lista compuesta por n elementos, ¿Qué esquema es el más adecuado?

Ramificación y poda, puesto que con buenas funciones de cota es más eficiente para este problema que vuelta atrás

Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante

\*Vuelta atrás, para este problema no hay un esquema más eficiente.

La complejidad en el menor de los casos de un algoritmo de ramificación y poda...

es siempre exponencial con el número de decisiones a tomar

suele ser polinómica con el número de alternativas por cada decisión

\*puede ser polinómica con el número de decisiones a tomar.

La complejidad en el peor de los casos de un algoritmo de ramificación y poda...

Puede ser exponencial con el número de alternativas por cada decisión

Puede ser polinómica con el número de decisiones a tomar

\*Es exponencial con el número de decisiones a tomar.

La estrategia de ramificación y poda genera las soluciones posibles al problema mediante...

\*un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones

un recorrido en profundidad del árbol que representa el espacio de soluciones

un recorrido en anchura que representa el espacio de soluciones.

¿Para qué sirven las cotas pesimistas en ramificación y poda?

Para tener la certeza de que la cota optimista está bien calculada

Para descartar nodos basándose en la preferencia por algún otro nodo ya completado

\*Para descartar nodos basándose en el beneficio esperado.

En los algoritmos de ramificación y poda, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir

\*En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir

No, nunca es así.

En los algoritmos de ramificación y poda, ¿el valor de una cota pesimista es menor que el valor de una cota optimista?

En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir

\*En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir

Sí, siempre es así.

En los algoritmos de ramificación y poda...

El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida

\*Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima

Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.

La ventaja de la estrategia ramificación y poda frente a vuelta atrás es que la primera genera las soluciones posibles al problema mediante...

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

\*las otras dos opciones son verdaderas

un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.

¿Cuál es la diferencia principal entre una solución de vuelta atrás y una solución de ramificación y poda para el problema de la mochila?

\*El orden de exploración de las soluciones

El coste asintótico en el caso peor

El hecho que la solución de ramificación y poda puede empezar con una solución subóptima voraz y la de vuelta atrás no.

Tratándose de un problema de optimización, en la lista de nodos vivos de ramificación y poda...

sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento

puede haber nodos que no son prometedores

\*las otras dos opciones son ciertas.

Cuando resolvemos un problema mediante un esquema de ramificación y poda...

\*los valores entre los cuales se elige en cada una de las decisiones tienen que formar un conjunto finito

las decisiones sólo pueden ser binarias

los valores entre los cuales se elige en cada una de las decisiones pueden formar un conjunto infinito.

La estrategia de ramificación y poda necesita cotas pesimistas...  
para decidir el orden de visita de los nodos del árbol de soluciones  
\*sólo si se usa para resolver problemas de optimización  
para determinar si una solución es factible.

El uso de funciones de cota en ramificación y poda...  
transforma en polinómicas complejidades que antes eran exponenciales  
\*puede reducir el número de instancias del problema que pertenecen al caso peor  
garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.

En la estrategia de ramificación y poda...  
\*cada nodo tiene su propia cota pesimista y también su propia cota optimista  
cada nodo tiene su propia cota pesimista, la cota optimista sin embargo, es común  
para todos los nodos  
cada nodo tiene su propia cota optimista, la cota pesimista sin embargo, es común  
para todos los nodos.

Si para resolver un mismo problema usamos un algoritmo de vuelta atrás y lo  
modificamos mínimamente para convertirlo en un algoritmo de ramificación y poda,  
¿qué cambiamos realmente?

La comprobación de las soluciones factibles: en ramificación y poda no es necesario  
puesto que sólo genera nodos factibles

Cambiamos la función que damos a la cota pesimista  
\*Aprovechamos mejor las cotas optimistas.

Si para resolver un mismo problema usamos un algoritmo de ramificación y poda y lo  
modificamos mínimamente para convertirlo en un algoritmo de vuelta atrás, ¿qué  
cambiamos realmente?

cambiamos la función que damos a la cota pesimista  
\*provocamos que las cotas optimistas pierdan eficacia  
sería necesario comprobar si las soluciones son factibles o no puesto que ramificación  
y poda solo genera nodos factibles.

En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

Las dos anteriores son verdaderas

garantizan que no se va a explorar nunca todo el espacio de soluciones posibles

\*pueden eliminar soluciones parciales que son factibles.

En ausencia de cotas optimista y pesimistas, la estrategia de vuelta atrás...

no se puede usar para resolver problemas de optimización

debe recorrer siempre todo el árbol

\*no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.

La estrategia de vuelta atrás es aplicable a problemas de selección y optimización en los que:

El espacio de soluciones puede ser tanto finito como infinito pero en este último caso debe ser al menos numerable

El espacio de soluciones es un conjunto infinito

\*El espacio de soluciones es un conjunto finito.

Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila:

El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico del objeto

El valor de una mochila

\*El valor óptimo de la mochila continua correspondiente.

Decid cuál de estas tres no sirve como cota optimista para obtener el valor óptimo de la mochila discreta:

El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido

\*El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos

El valor de la mochila continua correspondiente.

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

\*El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos

El valor de la mochila continua correspondiente

El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.

Es un problema de optimización, si el dominio de las decisiones es un conjunto infinito

Podemos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable

Es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo soluciones

\*Una estrategia voraz puede ser la única alternativa.

Dado un problema de optimización cualquiera, ¿la estrategia de vuelta atrás garantiza la solución óptima?

\*Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado

Sí, puesto que ese método analiza todas las posibilidades

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.

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. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta. ¿Qué tipo de cota sería?

\*Sería una cota pesimista siempre que se tenga la certeza de que esa aproximación encuentra la solución factible.

Ninguna de las otras dos opciones

Sería una cota optimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.

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. Como también se conocen las coordenadas geográficas de cada ciudad se quiere usar la distancia geográfica (en línea recta) entre cada par de ciudades para como cota para limitar la búsqueda en un algoritmo de vuelta atrás. ¿Qué tipo de cota sería?

No se trataría de ninguna poda puesto que es posible que esa heurística no encuentre una solución factible

Una cota pesimista

\*Una cota óptima.

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 geográfica (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. Seleccione una:

\*El nuevo algoritmo solo será más rápido para algunas instancias del problema

Esta estrategia no asegura que se obtenga el camino más corto

El nuevo algoritmo siempre sea más rápido.

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 geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino según su distancia geográfica. Este algoritmo voraz, ¿serviría como cota pesimista? Seleccione una:

\*No, ya que no asegura que se encuentre una solución factible

Sí, puesto que la distancia geográfica asegura que otra solución mejor no es posible

No, ya que en algunos casos puede dar distancias menores que la óptima.

Dado un problema de optimización cualquiera, ¿la estrategia de vuelta atrás garantiza la solución óptima?

Sí, puesto que ese método analiza todas las posibilidades.

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.

\*Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.

En los algoritmos de ramificación y poda

Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.

\*Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.

Una cota pesimista es el valor que a lo sumo alcanza cualquier nodo factible que no es el óptimo.

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.

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.

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.

\*Se puede mejorar la eficiencia del algoritmo convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.

Sí un problema de optimización lo es para una función que toma valores continuos ...

La programación dinámica iterativa siempre es mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.

\*La programación dinámica recursiva puede resultar mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.

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.

El uso de funciones de cota en ramificación y poda...

transforma en polinómicas complejidades que antes eran exponenciales.

garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.

\*puede reducir el número de instancias del problema que pertenecen al caso peor.

Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

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.

Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.

\*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.

¿Para cuál de estos problemas de optimización existe una solución voraz?

El problema de la mochila discreta.

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.

\*El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.

Se desea encontrar el camino más corto entre dos ciudades. Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela [por ejemplo, -1] si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geométrica (en línea recta) entre cada par de ciudades. Se pretende acelerar la búsqueda de un algoritmo de ramificación y poda priorizando los nodos vivos (ciudades) que estén a menor distancia geográfica de la ciudad objetivo.

\*El nuevo algoritmo no garantiza que vaya a ser más rápido para todas las instancias del problema posibles.

El nuevo algoritmo siempre sera más rápido.

Esta estrategia no asegura que se obtenga el camino mas corto.

La complejidad temporal en el mejor de los casos...

es el tiempo que tarda el algoritmo en resolver el problema de tamaño o talla más pequeña que se le puede presentar.

las otras dos opciones son ciertas.

\*es una función del tamaño o talla del problema que tiene que estar definida para todos los posibles valores de ésta.

La mejor solución que se conoce para el problema de la mochila continua sigue el esquema

...divide y vencerás.

...ramificación y poda.

\*...voraz.

La complejidad en el mejor de los casos de un algoritmo de ramificación y poda ...

es siempre exponencial con el número de decisiones a tomar.

\*puede ser polinómica con el número de decisiones a tomar.

suele ser polinómica con el número de alternativas por cada decisión.

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?

Que el algoritmo no encuentre ninguna solución.

\*Que se reconsidere la decisión ya tomada anteriormente respecto a la selección de un elemento anterior respecto a la selección de un elemento a la vista de la decisión que se debe tomar en el instante actual.

Que la solución no sea la óptima.

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

\*Programación dinámica.

Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.

El método voraz.

Sea la siguiente relación de recurrencia

¿en cuál de estos tres casos nos podemos encontrar?

Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in O(n^2)$ , ¿en cuál de estos tres casos nos podemos encontrar?

$g(n) = n$

\* $g(n) = n^2$

$g(n) = 1$ .

Un algoritmo recursivo basado en el esquema divide y vencerás ...

...nunca tendrá una complejidad exponencial.

\*...será más eficiente cuanto más equitativa sea la división en subproblemas.

Las dos anteriores son ciertas.

En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.

Las otras dos opciones son ciertas.

\*pueden eliminar soluciones parciales que son factibles.

Uno de estos tres problemas no tiene una solución eficiente que siga el esquema de programación dinámica

El Problema de la mochila discreta.

\*El Problema de las torres de Hanoi

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 el valor que se obtiene con el método voraz para el problema de la mochila discreta es...

... una cota inferior Para el valor óptimo, pero que nunca coincide con este.

... una cota superior para el valor óptimo.

\* ... una cota inferior Para el valor óptimo que a veces puede ser igual a este.

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

El valor de la mochila continua correspondiente

El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.

\*El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

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?

$O(2^n)$

\* $O(n^3)$

$O(n^2)$ .

El siguiente programa resuelve el problema de cortar un tubo de longitud  $n$  en segmentos de longitud entera entre 1 y  $n$  de manera que se maximice el precio de acuerdo con una tabla que da el precio para cada longitud, pero falta un trozo. ¿Qué debería ir en lugar de XXXXXXXX?

```

void fill(price 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;
}

```

p,r-1,n

p,r,n-r[n]

\*p,r,n-i.

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  $f(n) \in O(n^2)$

a

\*b

c.

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.

La complejidad temporal de la mejor solución posible al problema es  $O(n!)$ .

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.

\*La complejidad temporal de la mejor solución posible al problema es  $O(n^2)$ .

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).

Calcular la distancia geométrica (en línea recta) entre la ciudad origen y destino.

Para todas las ciudades que son alcanzables en un paso desde la ciudad inicial, sumar la distancia a dicha ciudad y la distancia geometrica hasta la ciudad destino.

\*Calcular la distancia recorrida moviéndose al azar por el grafo hasta llegar(por azar) a la ciudad destino.

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?

Cambiamos la función que damos a la cota pesimista.

\*El algoritmo puede aprovechar mejor las cotas optimistas.

La comprobación de las soluciones factibles: en ramificación y poda no es necesario puesto que sólo genera nodos factibles.

Dadas las siguientes funciones: Se quiere reducir la complejidad temporal de la función  $g$  usando programación dinámica iterativa. ¿cuál sería la complejidad espacial?

```
// 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());
}
```

cúbica

\*cuadrática

exponencial.

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?

$O(n^2)$

\* $O(n)$

$O(\sqrt{n})$ .

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)

No, nunca es así.

\*En general si, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.

En general si, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.

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?

Sí, tanto si se usan cotas optimistas para podar el árbol de búsqueda como si no.

No, ya que en cualquier caso se deben explorar todas las soluciones factibles.

\*Sí, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.

Garantiza el uso de una estrategia "divide y vencerás" la existencia de una solución de complejidad temporal polinómica a cualquier problema?

Sí, en cualquier caso.

\*No

Sí, pero siempre que la complejidad temporal conjunta de las operaciones de descomposición del problema y la combinación de las soluciones sea polinómica.

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...

es irrelevante si no se utilizan mecanismos de poda basados en la mejor solución hasta el momento.

puede ser relevante si se utilizan mecanismos de poda basados en estimaciones optimistas.

\*Las otras dos opciones son ciertas.

La versión de Quicksort que utiliza como pivote el elemento del vector que ocupa la primera posición ...

no presenta caso mejor y peor para instancias del mismo tamaño.

se comporta mejor cuando el vector ya está ordenado.

\*se comporta peor cuando el vector ya está ordenado.

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz (greedy) que es óptima?

\*El problema de la mochila discreta.

El problema de la mochila continua o con fraccionamiento.

El árbol de cobertura de coste mínimo de un grafo conexo.

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

\*una estrategia voraz puede ser la única alternativa.

es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione.

podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.

Dado un problema de optimización, el método voraz...

siempre obtiene la solución óptima.

\*garantiza la solución óptima sólo para determinados problemas.

siempre obtiene una solución factible.

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que...

en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.

El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.

\*en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos.

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?

Sí, ya que el mergesort es  $O(n \log n)$  y este es  $O(n)$

\*Sólo si  $d \log d > kn$  (donde  $k$  es una constante que depende de la implementación)

No, ya que este algoritmo ha de recorrer varias veces el vector de booleanos.

¿Cuál de estos problemas tiene una solución eficiente utilizando programación dinámica?

El problema de la asignación de tareas.

\*El problema del cambio.

La mochila discreta sin restricciones adicionales.

Dí cuál de estos tres algoritmos no es un algoritmo de "divide y vencerás"

Quicksort

Mergesort

\*El algoritmo de Prim.

Si  $f(n) \in O(n^3)$ , ¿puede pasar que  $f(n) \in O(n^2)$ ?

No, porque  $n^3 \neq O(n^2)$

Sólo para valores bajos de  $n$

\*Es perfectamente posible, ya que  $O(n^2) \subset O(n^3)$ .

## COMPLEJIDADES

¿Cuál es el objetivo de la etapa de análisis en el Diseño y Análisis de un Algoritmo?

Determinar el lenguaje y herramientas disponibles para su desarrollo.

\*Estimar los recursos que consumirá el algoritmo una vez implementado.

Estimar la potencia y características del equipo informático necesarios para el correcto funcionamiento del algoritmo.

¿Cuál de las siguientes jerarquías de complejidades es la correcta?

$O(1) \subset O(\lg n) \subset O(\lg \lg n) \subset \dots$

$\dots \subset (n!) \subset O(2^n) \subset O(n^n)$

\* $\dots \subset (2^n) \subset O(n!) \subset O(n^n)$ .

¿Cuál de los siguientes algoritmos de ordenación tiene menor complejidad?

Burbuja

Inserción directa

\*Mergesort.

¿El tiempo de ejecución de un algoritmo depende de la talla del problema?

Sí, siempre

No, nunca

\*No necesariamente.

Ordena de menor a mayor las siguientes complejidades:

- |                 |
|-----------------|
| 1. $O(1)$       |
| 2. $O(n^2)$     |
| 3. $O(n \lg n)$ |
| 4. $O(n!)$      |

3,1,2 y 4

\*1,3,2 y 4

1,3,4 y 2.

El estudio de la complejidad resulta realmente interesante para tamaños grandes de problema por varios motivos:

Las diferencias reales en tiempo de compilación de algoritmos con diferente coste para tamaños pequeños del problema no suelen ser muy significativas.

Las diferencias reales en tiempo de ejecución de algoritmos con diferente coste para tamaños grandes del problema no suelen ser muy significativas.

\*Ninguna de las anteriores.

¿Por que se emplean funciones de coste para expresar el coste de una algoritmo?

Para poder expresar el coste de los algoritmos con mayor exactitud

\*Para que la expresión del coste del algoritmo sea válida para cualquier entrada al mismo

Para poder expresar el coste de un algoritmo mediante una expresión matemática.

El caso base de una ecuación de recurrencia asociada a la complejidad temporal de un algoritmo expresa:

El coste de dicho algoritmo en el mejor de los casos.

El coste de dicho algoritmo en el peor de los casos.

\*Ninguna de las anteriores.

La complejidad de la función TB es:

```
función TB (A: vector[λ]; iz , de : N) : N
var n,i:N;
n=iz-de+1
opcion
 (n < 1) : devuelve (0);
 (n = 1) : devuelve (1);
 (n > 1) : si (A[iz] = A[de]) entonces
 devuelve (TB(A, iz + 1, de - 1) + 1);
 sino
 devuelve (TB(A, iz + 1, de - 1));
 finsi ;
fopcion
fin
```

\* $\Theta(n)$

$\Theta(n \cdot \lg n)$

$\Theta(n^2 \cdot \lg n)$ .

Dado el polinomio  $f(n) = A(m) n^m + A(m-1) n^{m-1} + \dots + A_0$ , con  $A(m) \in \mathbb{R}^+$  entonces  $f$  pertenece al orden:

$O(n^m)$ .

$\Omega(n^m)$ .

\*La dos respuestas anteriores son correctas.

Si  $f_1(n) \in O(g_1(n))$  y  $f_2(n) \in O(g_2(n))$  entonces:

$f_1(n) \cdot f_2(n) \in O(\max(g_1(n), g_2(n)))$

\* $f_1(n) \cdot f_2(n) \in O(g_1(n) \cdot g_2(n))$

Ambas son correctas.

Si  $f_1(n) \in O(g_1(n))$  y  $f_2(n) \in O(g_2(n))$  entonces:

$f_1(n) + f_2(n) \in O(\max(g_1(n), g_2(n)))$

$f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$

\*Ambas son correctas.

Un algoritmo cuya talla es  $n$  y que tarda  $40^n$  segundos en resolver cualquier instancia tiene una complejidad temporal:

$\Theta(n^n)$

\* $\Theta(4^n)$

Ninguna de las anteriores.

Si dos algoritmos tienen la misma complejidad asintótica:

\*No necesitan exactamente el mismo tiempo para su ejecución.

Necesitan exactamente el mismo tiempo para su ejecución.

Ninguna de las anteriores.

Los algoritmos directos de ordenación, respecto de los indirectos:

\*Presentan una mayor complejidad temporal y sus tiempos de ejecución absolutos son mayores.

Presentan una menor complejidad temporal y sus tiempos de ejecución absolutos son menores.

Presentan una mayor complejidad temporal si bien sus tiempos de ejecución absolutos son menores.

La talla o tamaño de un problema depende de:

Conjunto de valores asociados a la entrada y salida del problema.

Conjunto de valores asociados a la salida del problema.

\*Conjunto de valores asociados a la entrada del problema.

En un algoritmo recursivo, la forma de dividir el problema en subproblemas:

Influye en la complejidad espacial del mismo.

\*Influye en su complejidad temporal.

No influye en ninguna de sus complejidades.

$f(n) = 5n + 3m \cdot n + 11$  entonces  $f(n)$  pertenece a:

O ( $n \cdot m$ ).

O ( $n^m$ ).

\*Las dos son correctas.

El sumatorio, desde  $i=1$  hasta  $n$ , de  $i^k$  pertenece a:

\*O( $n^{(k+1)}$ )

O( $n^k$ )

Ninguna de las anteriores.

La complejidad de la función A2 es:

```

Funcion A2 (n, a: entero):entero;
Var r: entero; fvar
 si ($a^2 > n$) devuelve 0
 sino
 r:= A2(n, 2a);
 opción
 n < a^2 : devuelve r;
 n $\geq a^2$: devuelve r + a;
 fopción
 fsi
fin

```

$O(\sqrt{n} \cdot a)$

\* $O(\sqrt{n} / a)$

$O(n / \sqrt{a})$ .

Cual de las siguientes definiciones es cierta:

\*Las cotas de complejidad se emplean cuando para una misma talla se obtienen diferentes complejidades dependiendo de la entrada al problema.

Las cotas de complejidad se emplean cuando para diferentes tallas se obtienen diferentes complejidades dependiendo de la entrada al problema.

Ninguna de las anteriores.

Cuando para distintas instancias de problema con el mismo tamaño no obtenemos el mismo resultado:

No es posible calcular la complejidad a priori y debemos ejecutar el programa varias veces con la misma talla y obtener el tiempo medio para hallar la complejidad media.

No se puede aplicar la técnica de paso de programa, ya que esta técnica es para calcular la complejidad a priori.

\*Calculamos el máximo y mínimo coste que nos puede dar el algoritmo.

$f(n) = 5n+5$  ¿  $f(n)$  pertenece a  $O(n)$ ?

Sí. El valor de  $c$  es 5 y el valor mínimo de  $n_0$  es de 3

Sí. El valor de  $c$  es 9 y el valor mínimo de  $n_0$  es de 1

\*Sí. El valor de c es 6 y el valor mínimo de n<sub>0</sub> es de 5.

f(n) = 10n+7 ¿ f(n) pertenece a O(n<sup>2</sup>)?

Sí. Para c = 1 y a partir de un valor de n<sub>0</sub> = 10.

\*Sí Para cualquier valor de c positivo siempre existe un n<sub>0</sub> a partir del que se cumple.

No.

Si f(n) ∈ Ω(g(n)) entonces:

\* $\exists c, n_0 \in \mathbb{R}^+: f(n) \geq c \cdot g(n) \forall n \geq n_0$

$\exists c, n_0 \in \mathbb{R}^+: f(n) \geq c \cdot g(n) \forall n$

$\exists c, n_0 \in \mathbb{R}^+: f(n) \leq c \cdot g(n) \forall n \geq n_0$ .

El coste asociado a la siguiente ecuación de recurrencia es:

$$f(n) \begin{cases} 1 & n \leq 1 \\ n + f(n/2) + f(n/2) & n > 1 \end{cases}$$

$\Theta(n \lg(n^2))$

$\Theta(n^2 \lg(n))$

\* $\Theta(n \lg(n))$ .

En los algoritmos de ramificación y poda, ¿el valor de una cota pesimista es siempre menor o igual que el valor de una cota optimista? (se entiende que ambas cotas se aplican sobre el mismo nodo)

\*Solo si se trata de un problema de maximización

No, depende del problema

Sólo si se trata de un problema de minimización.

¿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?

\*La división del problema en subproblemas

La combinación de las soluciones a los subproblemas

El número de llamadas recursivas que se hacen.

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es falsa:

Si  $g(n) \in O(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

\*Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.

Si  $g(n) \in O(n)$  la relación de recurrencia representa la complejidad temporal en el mejor caso del algoritmo de búsqueda del  $k$ -ésimo elemento más pequeño de un vector (estudiado en clase).

¿Cuál es el coste temporal asintótico de la siguiente función?

¿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)  $O(n \log n)$
- (b)  $O(n)$
- (c)  $O(n^2)$

\*b

c.

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.

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.

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.

\*Se puede mejorar la eficiencia del algoritmo convirtiendo el algoritmo recursivo directamente en iterativo sin cambiar su funcionamiento básico.

¿Garantiza la estrategia "divide y vencerás" una solución de complejidad temporal polinómica a cualquier problema?

Sí, en cualquier caso.

\*No, la complejidad temporal puede ser incluso peor que cualquier función polinómica.

Sí, pero siempre que la complejidad temporal conjunta de las operaciones de descomposición del problema y la combinación de las soluciones sea polinómica.

¿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?

Para este problema, las dos estrategias citadas serían similares en cuanto a eficiencia

La estrategia divide y vencerás

\*Programación dinámica.

Con respecto al tamaño del problema, ¿Cuál es la complejidad temporal de la siguiente función?

---

```

unsigned f(unsigned n) {
 if(n < 2) return n;
 unsigned sum=0;
 for(int i = 0; i < 10; i++)
 sum+=n;
 return sum;
}

```

---

$\Theta(n)$

\* $\Theta(1)$

Ninguna de las otras dos opciones es correcta.

¿Qué aporta la técnica Ramificación y poda frente a Vuelta atrás?

Eficiencia, los algoritmos de ramificación y poda son más eficientes que los de vuelta atrás...

\*La posibilidad de analizar distintas estrategias para seleccionar el siguiente nodo a expandir.

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.

Si  $\lim \dots$  entonces.

. Si  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$  entonces ...

---

...  $g(n) \in O(f(n))$

\*...  $f(n) \in O(g(n))$

...  $f(n) \in \Theta(g(n))$ .

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?

Sí, tanto si se usan cotas optimistas para podar el árbol de búsqueda como si no.

\*Sí, pero sólo si se usan cotas optimistas para podar el árbol de búsqueda.

No, ya que en cualquier caso se deben explorar todas las soluciones factibles.

En un algoritmo de ramificación y poda, si la lista de nodos vivos no está ordenada de forma apropiada...

...podría ocurrir que se pade el nodo que conduce a la solución óptima.

\*...podría ocurrir que se exploren nodos de forma innecesaria.

...podría ocurrir que se descarten nodos factibles.

El algoritmo de ordenación Quicksort divide el problema en dos subproblemas. ¿Cuál es la complejidad temporal asintótica de realizar esa división?

\* $O(n)$

$O(\log n)$

$O(n \log n)$ .

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?

Que la solución no sea la óptima.

\*Que se reconsidera la decisión ya tomada anteriormente respecto a la selección de un elemento a la vista de la decisión que se debe tomar en un instante.

Que el algoritmo no encuentre ninguna solución.

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;
}

```

$\Omega(n)$  y  $O(n^2)$

$\Theta(2^n)$

\* $\Omega(n)$  y  $O(2^n)$ .

¿Para cuál de estos problemas de optimización existe una solución voraz?

El problema de la mochila discreta.

\*El árbol de recubrimiento mínimo para un grafo no dirigido con pesos

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.

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?

\*Una cota optimista.

No se trataría de ninguna cota puesto que es posible que esa heurística no encuentre una solución factible.

Una cota pesimista.

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

\*El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

El valor de la mochila continua correspondiente.

El valor de una mochila que contiene todos los objetos restantes aunque se pase del peso máximo permitido.

En los algoritmos de ramificación y poda...

Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.

\*Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.

Una cota pesimista es el valor que a lo sumo alcanza cualquier nodo factible que no es el óptimo.

El uso de funciones de cota en ramificación y poda...

\*...puede reducir el número de instancias del problema que pertenecen al caso peor.

...transforma en polinómicas complejidades que antes eran exponenciales.

...garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.

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]-1,p
(b) n-i,m,p
(c) n-m[n],m,p

```

n,m[n]-1,p

\*n-i,m,p

n-m[n],m,p

Sea la siguiente relación de recurrencia: Si  $T(n) \in O(n)$ , ¿en cuál de estos tres casos nos podemos encontrar?

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

\* $g(n) = 1$

$g(n) = n$

$g(n) = n^2$ .

¿Se puede reducir el coste temporal de un algoritmo recursivo almacenando los resultados devueltos por las llamadas recursivas?

No, sólo se puede reducir el coste convirtiendo el algoritmo recursivo en iterativo

\*Si, si se repiten llamadas a la función con los mismos argumentos

No, ello no reduce el coste temporal ya que las llamadas recursivas se deben realizar de cualquier manera.

Si  $f(n) \in O(n^2)$ , ¿podemos decir siempre que  $f(n) \in O(n^3)$ ?

**Si  $f(n) \in \Theta(n^2)$ , ¿podemos decir siempre que  $f(n) \in O(n^3)$ ?**

- (a) Sí, ya que  $n^2 \in \Omega(n^3)$
- (b) Sí, ya que  $n^2 \in O(n^3)$
- (c) No, ya que  $n^2 \notin O(n^3)$

a

\*b

c.

Con respecto al parámetro n, ¿Cuál es la complejidad temporal de la siguiente función?

**Con respecto al parámetro n, ¿Cuál es la complejidad temporal de la siguiente función?**

```
void f(unsigned n) {
 if(n < 2) return;
 for(int i = 0; i < pow(n, 3); i++)
 cout << "*";
 f(n / 2);
}
```

- (a)  $\Theta(n^4)$
- (b)  $\Theta(n^3 \log n)$
- (c)  $\Theta(n^3)$

a

b

\*c.

Si un problema de optimización lo es para una función que toma valores continuos...

La programación dinámica iterativa siempre es mucho más eficiente que la programación dinámica recursiva en cuanto al uso de memoria.

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.

\*La programación dinámica recursiva puede resultar mucho más eficiente que la programación dinámica iterativa en cuanto al uso de memoria.

En un algoritmo de ramificación y poda, el orden escogido para priorizar los nodos en la lista de nodos vivos . . .

...determina la complejidad temporal en el peor de los casos del algoritmo.

...nunca afecta al tiempo necesario para encontrar la solución óptima.

\*...puede influir en el número de nodos que se descartan sin llegar a expandirlos.

Se pretende resolver un problema de maximización utilizando ramificación y poda y para ello se dispone de tres posibles cotas optimistas. ¿Cuál deberíamos utilizar para tratar de reducir la cantidad de nodos a explorar?

La que obtenga valores más elevados.

\*La que obtenga valores más pequeños.

La que más se acerque a una heurística voraz que obtenga una solución aproximada del problema.

Tratándose de un esquema general para resolver problemas de minimización, ¿qué falta en el hueco?

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
```

\*n.optimistic\_b() <= pb

n.optimistic\_b() >= pb

n.pesimistic\_b() <= pb

Sea  $g(n) = \text{sumatorio}$ . Di cuál de las siguientes afirmaciones es falsa:

Sea  $g(n) = \sum_{i=0}^K a_i n^i$ . Di cuál de las siguientes afirmaciones es falsa:

- (a)  $g(n) \in \Omega(n^K)$
- (b)  $g(n) \in \Theta(n^K)$
- (c) Las otras dos afirmaciones son ambas falsas.

a

b

\*C.

Dadas las siguientes funciones:

f

g

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, 0, v.size());
}
```

Se quiere reducir la complejidad temporal de la función g usando programación dinámica iterativa. ¿cuál sería la complejidad espacial?

(a)

Se quiere reducir la complejidad temporal de la función g usando programación dinámica iterativa. ¿cuál sería la complejidad espacial?

cúbica

\*cuadrática

exponencial

¿Cuál es la definición correcta de  $O(g)$ ?

¿Cuál es la definición correcta de  $O(g)$ ?

- (a)  $O(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ | \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \leq cf(n)\}$
- (b)  $O(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ | \exists c \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \leq cg(n)\}$
- (c)  $O(g) = \{f : \mathbb{N} \rightarrow \mathbb{R}^+ | \forall c \in \mathbb{R}^+, \forall n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \leq cf(n)\}$

a

\*b

c.

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...

es irrelevante si no se utilizan mecanismos de poda basados en la mejor solución hasta el momento.

\*las dos anteriores son ciertas.

puede ser relevante si se utilizan mecanismos de poda basados en estimaciones optimistas.

¿Cuál es la diferencia principal entre una solución de vuelta atrás y una solución de ramificación y poda para el problema de la mochila?

El coste asintótico en el caso peor.

El hecho que la solución de ramificación y poda puede empezar con una solución subóptima voraz y la de vuelta atrás no.

\*El orden de exploración de las soluciones.

La versión de Quicksort que utiliza como pivote el primer elemento del vector...

... no presenta caso mejor y peor para instancias del mismo tamaño.

... se comporta mejor cuando el vector ya está ordenado.

\*... se comporta peor cuando el vector ya está ordenado.

Llamando  $n$  al tamaño del problema, ¿Cuál es la complejidad temporal de la siguiente función?

Llamando  $n$  al tamaño del problema, ¿Cuál es la complejidad temporal de la siguiente función?

```
/* Llamada inicial: g(v, 0) */
void g(vector <unsigned> &v, unsigned j) {
 if (j==v.size()) return;
 for (unsigned i=j; i < v.size(); i++){
 swap(v[j], v[i]);
 g(v, j+1);
 swap(v[j], v[i]);
 }
}
```

- (a)  $O(n^n)$
- (b)  $O(n)$
- (c)  $O(n!)$

a

b

\*c.

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.

La complejidad temporal de la mejor solución posible al problema está en  $\Omega(n^2)$

\*La complejidad temporal de la mejor solución posible al problema es  $O(n \log n)$

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.

Tenemos un vector ordenado y queremos comprobar si contiene un elemento dado. ¿Cuál sería la complejidad temporal mas ajustada para hacerlo?

El tamaño del vector.

Constante con el tamaño del vector.

\*El logaritmo del tamaño del vector.

En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

\*pueden eliminar soluciones parciales que son factibles.

las dos anteriores son verdaderas.

garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.

Los algoritmos de vuelta atrás que hacen uso de cotas optimistas generan las soluciones posibles al problema mediante...

... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.

... 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.

\* ... un recorrido en profundidad del árbol que representa el espacio de soluciones.

Dado el problema del laberinto con tres movimientos, ¿se puede aplicar un esquema de programación dinámica para obtener un camino de salida?

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.

\*Si, en caso de existir con este esquema siempre se puede encontrar un camino de salida

No, para garantizar que se encuentra un camino de salida hay que aplicar métodos de búsqueda exhaustiva como vuelta atrás o ramificación y poda.

En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...

... debe recorrer siempre todo el árbol.

... no se puede usar para resolver problemas de optimización.

\*... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.

Decid cuál de estas tres es la cota optimista más ajustada al valor óptimo de la mochila discreta:

El valor de una mochila que contiene todas los objetos aunque se pase del peso máximo permitido.

\*El valor de la mochila continua correspondiente.

El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

El coste temporal asintótico de insertar un elemento en un vector ordenado de forma que continue ordenado es:

$\Omega(n^2)$

\* $O(n)$

$O(\log n)$ .

¿En ramificación y poda, tiene sentido utilizar la cota optimista de los nodos como criterio para ordenar la lista de nodos vivos?

Sí, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista.

No, la cota optimista sólo se utiliza para determinar si una n-tupla es prometedora.

\*Sí, aunque no es una garantía de que sea una buena estrategia de búsqueda.

¿Qué estrategia de búsqueda es a priori más apropiada en un esquema de vuelta atrás?

Explorar primera los nodos que están más completados.

\*En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.

Explorar primero los nodos con mejor cota optimista.

Sea la siguiente relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in O(n)$ , ¿en cuál de estos tres casos nos podemos encontrar?

Si  $T(n) \in O(n)$ , ¿en cuál de estos tres casos nos podemos encontrar?

\*Las otras dos opciones son ambas ciertas

$g(n) = \log n$

$g(n) = \sqrt{n}$ .

Si  $f \in \Omega(g_1)$  y  $f \in \Omega(g_2)$  entonces:

\* $f \in \Omega(g_1+g_2)$

$f \notin \Omega(\min(g_1, g_2))$

$f \in \Omega(g_1 * g_2)$ .

Dada la siguiente función:

Int exa

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?

$O(n^2)$

\* $O(n)$

$O(\log n)$ .

El esquema de vuelta atrás...

Se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado.

Las otras dos opciones son ambas verdaderas.

\*Garantiza que encuentra la solución Óptima & cualquier problema de selección discreta.

En el esquema de ramificación y poda, ¿qué estructura es la más adecuada si queremos realizar una exploración por niveles?

Cola de prioridad

Pila

\*Cola.

¿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?

\*Nada de interés

El coste temporal promedio

El coste temporal asintótico en el caso medio.

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?

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.

\*De una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización.

De una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización.

¿Qué complejidad se obtiene a partir de la relación de recurrencia  $T(n) = 8T(n/2) + n^3$  con  $T(1) = O(1)$ ?

\* $O(n^3 \log n)$

$O(n^3)$

$O(n \log n)$ .

Dada la siguiente función:

Int exa

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;
}
```

La complejidad temporal exacta es  $\Theta(n^2)$

\*La complejidad temporal en el mejor de los casos es  $\Omega(n)$

La complejidad temporal en el mejor de los casos es  $\Omega(1)$ .

El esquema voraz...

\*Las otras dos opciones son ambas falsas.

Garantiza encontrar una solución a cualquier problema, aunque puede que no sea óptima.

Puede que no encuentre una solución pero si lo hace se garantiza que es óptima.

Cuando la descomposición de un problema de lugar a subproblemas de tamaño similar al original, muchos de los cuales se repiten, ¿qué esquema es a priori más apropiado?

Ramificación y poda.

Divide y vencerás.

\*Programación dinámica.

Dada la siguiente función (donde  $\max(a,b)$  )

## Float exa

---

Dada la siguiente función (donde  $\max(a, b) \in \Theta(1)$ ):

```
float exa(vector<float>&v, vector<int>&p, int P, int i)
{
 float a, b;
 if (i >= 0){
 if (p[i] <= P)
 a = v[i] + exa(v, p, P - p[i], i - 1);
 else a = 0;
 b = exa(v, p, P, i - 1);
 return max(a, b);
 }
 return 0;
}
```

La complejidad temporal en el peor de los casos es  $O(n^2)$

\*La complejidad temporal en el peor de los casos es  $O(2^n)$

La complejidad temporal en el mejor de los casos es  $\Omega(n^2)$ .

¿Cuál sería la complejidad temporal de la siguiente función tras aplicar programación dinámica?

## Double f

---

¿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);
}
```

---

\* $\Theta(n)$

$\Theta(n^2)$

$\Theta(n \times m)$ .

Dado el problema del laberinto con tres movimientos, se pretende conocer la longitud del camino de salida más corto. Para ella 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?

Que es un algoritmo voraz pero sin garantía de solucionar el problema...

\*Que en realidad no es un algoritmo voraz pues el criterio de selección no lo es.

Que en realidad no es un algoritmo voraz pues las decisiones que se toman no deberían reconsiderarse.

Si el coste temporal de un algoritmo es  $T(n)$ , ¿cuál de las siguientes situaciones es imposible?

\* $T(n) \in \Theta(n)$  y  $T(n) \in \Omega(n^2)$

$T(n) \in O(n)$  y  $T(n) \in \Theta(n)$

$T(n) \in \Omega(n)$  y  $T(n) \in \Theta(n^2)$ .

Se desea resolver el problema de la potencia enésima ( $x^n$ ), asumiendo que  $n$  es par y que se utilizará la siguiente recurrencia:  $\text{pot.}(x, n) = \text{pot.}(x, n/2) * \text{pot.}(x, n/2)$ ; ¿Qué esquema resulta ser más eficiente en cuanto al coste temporal?

En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.

\*Programación dinámica.

Divide y vencerás.

De las siguientes afirmaciones marca la que es verdadera.

El esquema de vuelta atrás no es compatible con el uso conjunto de cotas pesimistas y optimistas.

Las cotas pesimistas no son compatibles con un esquema de vuelta atrás.

\*En un esquema de vuelta atrás, las cotas pesimistas no tienen sentido si lo que se pretende es obtener todas las soluciones factibles.

Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿qué propiedad cumple una cota optimista?

Asegura un ahorro en la comprobación de todas las soluciones factibles.

Siempre es mayor o igual que la mejor solución pasible alcanzada.

\*Las otras dos opciones son ambas falsas.

Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial (1,1) hasta la casilla (n, m) y para ello se aplica el esquema programación dinámica para obtener un algoritmo lo más eficiente posible en cuanto a complejidad temporal y espacial. ¿Cuáles serían ambas complejidades?

Temporal  $\Theta(n \times m)$  y espacial  $\Theta(n \times m)$

\*Temporal  $\Theta(n \times m)$  y espacial  $\Theta(\min\{n, m\})$

Temporal  $\Theta(\max\{n, m\})$  y espacial  $\Theta(\max\{n, m\})$ .

¿Qué se deduce de  $f(n)$  y  $g(n)$  si se cumple  $\lim$ ?

¿Qué se deduce de  $f(n)$  y  $g(n)$  si se cumple  $\lim_{n \rightarrow \infty} (f(n)/g(n)) = k$ , con  $k \neq 0$ ?

\* $f(n) \in O(g(n))$  y  $g(n) \in O(f(n))$

$f(n) \in O(g(n))$  pero  $g(n) \notin O(f(n))$

$g(n) \in O(f(n))$  pero  $f(n) \notin O(g(n))$ .

Se desea obtener todas las permutaciones de una lista compuesta por  $n$  elementos.

¿Qué esquema es el más adecuado?

\*Vuelta atrás, es el esquema más eficiente para este problema.

Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.

Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.

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?

$$*nc(n, m) = nc(n - 1, m) + nc(n, m - 1) + nc(n - 1, m - 1)$$

$$nc(n, m) = nc(n - 1, m) * nc(n, m - 1) * nc(n - 1, m - 1)$$

Ninguna de las otras dos recurrencias se corresponde con un esquema de divide y vencerás.

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc... Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Se quiere resolver el problema mediante vuelta atrás. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?

Una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.

\*Un vector de booleanos.

Un par de enteros que indiquen los cortes realizados y el valor acumulado.

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.

\* $\Theta(n) \subset \Theta(n^2)$

$n + n \log n \in \Omega(n)$

$O(2^{(\log n)}) \subset O(n^2)$ .

En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...

Las otras dos opciones son ambas verdaderas...

\*Pueden eliminar vectores que representan posibles soluciones factibles.

Garantizan que no se va a explorar todo el espacio de soluciones posibles.

En el problema del viajante de comercio (travelling salesman problem) queremos listar todas las soluciones factibles. Lo más importante es conseguir una cota pesimista adecuada.

Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.

La 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.

\*El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.

Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?

Que se podría explorar menos nodos de los necesarios.

Que se podría podar el nodo que conduce a la solución óptima.

\*Que se podría explorar más nodos de los necesarios.

Un algoritmo recursivo basado en el esquema divide y vencerás...

Las otras dos opciones son ambas verdaderas.

\*... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a.

... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.

Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones de recurrencia expresa mejor su complejidad temporal para el caso general, siendo  $n$  el número de discos?

\* $T(n) = 2T(n - 1) + 1$

$T(n) = 2T(n - 1) + n$

$T(n) = T(n - 1) + n.$

Una de las prácticas de laboratorio consistió en el cálculo empírico de la complejidad temporal promedio del algoritmo de ordenación de vectores Quicksort tomando como centinela el elemento del vector que ocupa la posición central. ¿Cuál es el orden de complejidad que se obtuvo?

$n \log^2$

$n n^2$

\* $n \log n.$

Se desea ordenar una lista enlazada de  $n$  elementos haciendo uso del algoritmo Mergesort. En este caso, al tratarse de una lista, la complejidad temporal asintótica de realizar la división en subproblemas resulta ser lineal con el tamaño de esa lista. ¿Cuál sería entonces el coste temporal de realizar dicha ordenación?

\* $\Theta(n \log n)$

$\Theta(n^2)$

Ninguna de las otras dos opciones es cierta.

Dada la siguiente función:

Int exa

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.

La complejidad temporal exacta es  $\Theta(n \log n)$

La complejidad temporal en el mejor de los casos es  $\Omega(1)$

\*La complejidad temporal en el mejor de los casos es  $\Omega(n)$ .

Dado el problema del laberinto con tres movimientos, ¿cuál de las estrategias siguientes proveería de una cota optimista para ramificación y poda?

Suponer que ya no se van a realizar más movimientos.

\*Las otras dos estrategias son ambas válidas

Suponer que en adelante todas las casillas del laberinto son accesibles.

¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?

Que el algoritmo sería más lento pues se explorarían más nodos de los necesarios.

Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.

\*Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.

Los algoritmos de ordenación Quicksort y Mergesort tienen en común...

... que ordenan el vector sin usar espacio adicional.

... que se ejecutan en tiempo  $O(n)$ .

\* ... que aplican la estrategia de divide y vencerás.

Tenemos un conjunto de  $n$  enteros positivos y queremos encontrar el subconjunto de tamaño  $m$  de suma mínima...

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.

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.

\*Una técnica voraz daría una solución óptima.

Di cuál de estos resultados de coste temporal asintótico es falsa:

La ordenación de un vector usando el algoritmo Quicksort requiere en el peor caso un tiempo en  $\Omega(n^2)$

\*La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en  $\Omega(n^2)$

La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en  $O(\log n)$ .

La complejidad temporal (o coste temporal asintótico) en el mejor de los casos...

\*es una función de la talla, o tamaño del problema, que tiene que estar definida para todos los posibles valores de ésta.

es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar

las dos anteriores son verdaderas.

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.

\*No hay ningún problema en usar una técnica de vuelta atrás.

No se puede usar vuelta atrás porque las decisiones no son valores discretos.

No se puede usar vuelta atrás porque el número de combinaciones es infinito.

Dado un problema de optimización, ¿cuándo se puede aplicar el método de vuelta atrás?

\*Es condición necesaria (aunque no suficiente) que el dominio de las decisiones sea discreto o discretizable.

Es condición necesaria y suficiente que el dominio de las decisiones sea discreto o discretizable.

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.

¿Cuál de estas tres expresiones es cierta?

\* $O(2^{(\log n)}) \subset O(n^2) \subset O(2^n)$

$O(n^2) \subset O(2^{(\log n)}) \subseteq O(2^n)$

$O(n^2) \subset O(2^{(\log n)}) \subset O(2^n).$

Sea  $f(n)$  la solución de la relación de recurrencia  $f(n) = 2f(n/2) + n$ ;  $f(1) = 1$ . Indicad cuál de estas tres expresiones es cierta:

$f(n) \in \Theta(n^2)$

\* $f(n) \in \Theta(n \log n)$

$f(n) \in \Theta(n).$

Indicad cuál de estas tres expresiones es falsa:

$\Theta(n/2) = \Theta(n)$

$\Theta(n) \subset O(n)$

\* $\Theta(n) \subset \Theta(n^2).$

Indicad cuál es el coste temporal asintótico (o complejidad temporal), en función de  $n$ , del programa siguiente:

```
s=0; for(i=0; i<n; i++) for(j=i; j<n; j++) s+=n*i*j;
```

Es  $O(n^2)$  pero no  $\Omega(n^2)$

\*Es  $\Theta(n^2)$

Es  $\Theta(n)$ .

¿Pertenece  $3n^2 + 3$  a  $O(n^3)$ ?

\*Sí

No

Sólo para  $c = 1$  y  $n_0 = 5$ .

Las relaciones de recurrencia...

aparecen sólo cuando la solución sea del tipo divide y vencerás

\*expresan recursivamente el coste temporal de un algoritmo

sirven para reducir el coste temporal de una solución cuando es prohibitivo.

El coste temporal de un algoritmo se ajusta a la siguiente ecuación de recurrencia:

El coste temporal de un algoritmo se ajusta a la siguiente ecuación de recurrencia:

$$T(n) = \begin{cases} 1 & n = 0 \\ n + \sum_{j=0}^{n-1} T(j) & n > 1 \end{cases}$$

¿qué coste temporal asintótico (o complejidad temporal) tendrá el algoritmo?

¿qué coste temporal asintótico (o complejidad temporal) tendrá el algoritmo?

$O(n \log n)$

$O(n^2)$

\* $O(2^n)$ .

¿Cuál es el coste espacial asintótico del siguiente algoritmo?

¿Cuál es el coste espacial asintótico del siguiente algoritmo?

```
int f(int n) {
 int a = 1, r = 0;
 for(int i = 0; i . . .
 z = a + r;
 a = 2*r;
 }
 return r;
}
```

\*O(1)

O(log n)

O(n).

¿Qué algoritmo es asintóticamente más rápido, Quicksort o Mergesort?

Como su nombre indica, el Quicksort

\*Son los dos igual de rápidos, ya que el coste temporal asintótico de ambos es O( $n \log n$ )

El Mergesort es siempre más rápido o igual (salvo una constante) que el Quicksort.

El coste temporal del algoritmo de ordenación por inserción es...

\*O( $n^2$ )

O(n)

O( $n \log n$ ).

El problema de la función compuesta mínima consiste en encontrar, a partir de un conjunto de funciones dadas, la secuencia mínima de composiciones de éstas que permita transformar un número  $n$  en otro  $m$ . Se quiere resolver mediante ramificación y poda. ¿Cuál sería la forma más adecuada de representar las posibles soluciones?

Mediante un vector de booleanos

Mediante un vector de reales

\*Este problema no se puede resolver usando ramificación y poda si no se fija una cota superior al número total de aplicaciones de funciones.

¿Cuál de estas expresiones es falsa?

$2n^2 + 3n + 1 \in O(n^3)$

$n + n \log n \in \Omega(n)$

\* $n + n \log n \in \Theta(n)$ .

Un algoritmo recursivo basado en el esquema divide y vencerás...

\*Será más eficiente cuanto más equitativa sea la división en subproblemas

Las demás opciones son verdaderas

Nunca tendrá una complejidad exponencial.

El algoritmo Quicksort divide el problema en dos subproblemas. ¿Cuál es la complejidad temporal asintótica de realizar esa división?

$O(n \log n)$

$\Omega(n) \text{ y } O(n^2)$

\* $O(n)$ .

Uno de estos tres problemas no tiene una solución trivial y eficiente que siga el esquema voraz

\*El problema del cambio

El problema de la mochila discreta sin limitación en la carga máxima de la mochila

El problema de la mochila continua.

Estudiad la relación de recurrencia:

. Estudiad la relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT\left(\frac{n}{q}\right) + g(n) & \text{en otro caso} \end{cases}$$

(donde  $p$  y  $q$  son enteros mayores que 1). Di cuál de los siguientes esquemas algorítmicos produce de manera natural relaciones de recurrencia así.

(donde  $p$  y  $q$  son enteros mayores que 1). Di cuál de los siguientes esquemas algorítmicos produce de manera natural relaciones de recurrencia así.

\*Divide y vencerás

Ramificación y poda

Programación dinámica.

Sea  $g(n) = \text{sumatorio}$ . Di cuál de las siguientes afirmaciones es falsa:

Sea  $g(n) = \sum_{i=0}^K a_i n^i$ . Di cuál de las siguientes afirmaciones es falsa:

\*Las otras dos afirmaciones son falsas.

$g(n) \in \Theta(n^k)$

$g(n) \in \Omega(n^k)$ .

Si  $\lim = 0$  entonces...

Si  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$  entonces ...

$f(n) \in \Theta(g(n))$

\* $f(n) \in O(g(n))$

$g(n) \in O(f(n))$ .

¿Cuál es la diferencia principal entre una solución de vuelta atrás y una solución de ramificación y poda para el problema de la mochila?

El hecho que la solución de ramificación y poda puede empezar con una solución subóptima voraz y la de vuelta atrás no.

El coste asintótico en el caso peor

\*El orden de exploración de las soluciones.

Sea la siguiente relación de recurrencia

Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in O(n)$ , ¿en cuál de estos tres casos nos podemos encontrar?

Si  $T(n)$ , ¿en cuál de estos tres casos nos podemos encontrar?

$g(n) = n^2$

$g(n) = n$

\* $g(n) = 1$ .

Los algoritmos de vuelta atrás que hacen uso de cotas optimistas generan soluciones posibles al problema mediante...

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

un recorrido guiado por estimaciones de las mejores ramas del árbol que representan el espacio de soluciones

\*un recorrido en profundidad del árbol que representa el espacio de soluciones.

¿Se puede reducir el coste temporal de un algoritmo recursivo almacenando los resultados devueltos por las llamadas recursivas?

\*Sí, si se repiten llamadas a la función con los mismos argumentos

No, ello no reduce el coste temporal ya que las llamadas recursivas se deben realizar de cualquier manera

No, sólo se puede reducir el coste convirtiendo el algoritmo recursivo en iterativo.

La versión Quicksort que utiliza como pivote el elemento del vector que ocupa la posición central...

\*se comporta mejor cuando el vector está ordenado

no presenta caso mejor y peor para instancias del mismo tamaño

se comporta peor cuando el vector ya está ordenado.

La complejidad temporal de la solución de vuelta atrás al problema de la mochila discreta es...

cuadrática en el caso peor

exponencial en cualquier caso

\*exponencial en el caso peor.

Ante un problema de optimización resuelto mediante backtracking, ¿Puede ocurrir que el uso de las cotas pesimistas y optimistas sea inútil, incluso perjudicial?

Según el tipo de cota, las pesimistas puede que no descarten ningún nodo pero el uso de cotas optimistas garantiza la reducción del espacio de búsqueda

No, las cotas tanto optimistas como pesimistas garantizan la reducción del espacio de soluciones y por tanto la eficiencia del algoritmo

\*Sí, puesto que es posible que a pesar de utilizar dichas cotas no se descarte ningún nodo.

¿Qué se entiende por "tamaño del problema"?

El valor máximo que puede tomar una instancia cualquiera de ese problema

El número de parámetros que componen el problema

\*La cantidad de espacio en memoria que se necesita para codificar una instancia de ese problema.

Si  $\lim$ , ¿cuál de estas tres afirmaciones es falsa?

Si  $\lim_{n \rightarrow \infty} (f(n)/n^2) = k$ , y  $k \neq 0$ , ¿cuál de estas tres afirmaciones es falsa?

$f(n) \in O(n^3)$

\* $f(n) \in \Theta(n^3)$

$f(n) \in \Theta(n^2)$ .

La función gamma de un número semientero positivo (un número es semientero si al restarle 0.5 es entero) se define como:

La función  $\gamma$  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?

¿Se puede calcular usando programación dinámica iterativa?

No, ya que el índice del almacén sería un número real y no entero

No, ya que no podríamos almacenar los resultados intermedios en el almacén

\*Sí, pero la complejidad temporal no mejora.

¿Cuál de estas afirmaciones es falsa?

La solución de programación dinámica iterativa al problema de la mochila discreta realiza cálculos innecesarios

\*Los algoritmos iterativos de programación dinámica utilizan memoización para evitar resolver de nuevo los mismos subproblemas que se vuelven a presentar

La memoización evita que un algoritmo recursivo ingenuo resuelva repetidamente el mismo problema.

¿Cuál de estas afirmaciones es falsa?

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

\*Todos los problemas de optimización tienen una solución voraz que es óptima sea cual sea la instancia a resolver

Hay problemas de optimización para los cuales se puede obtener siempre la solución óptima utilizando una estrategia voraz.

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.

$$O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$$

$$(4^{\log_2(n)}) \subseteq O(n^2) \subset O(2^n)$$

$$O(2^{\log_2(n)}) \subseteq O(n^2) \subset O(n!)$$

\*a

b

c.

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.

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  $O(f) = \Omega(f) \cap \Theta(f)$

\*a

b

c.

Dada la relación de recurrencia:

Dada la relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT\left(\frac{n}{a}\right) + g(n) & \text{en otro caso} \end{cases}$$

(donde  $p$  y  $a$  son enteros mayores que 1 y  $g(n) = n^k$ ), ¿qué tiene que ocurrir para que se cumpla  $T(n) \in \Theta(n^k)$ ?

(donde p y a son enteros mayores que 1 y  $g(n) = n^k$ ), ¿qué tiene que ocurrir para que se cumpla  $T(n)$ ?

$p > a^k$

$p = a^k$

\* $p < a^k$ .

¿Cuál es la complejidad temporal en el mejor de los casos de la siguiente función?

Void examen

¿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;
 }
 }
 }
}
```

\* $\Omega(n)$

$\Omega(1)$

Esta función no tiene caso mejor.

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...

se van a utilizar tantos colores distintos a los ya utilizados como vértices quedan por colorear

\*no se van a utilizar colores distintos a los ya utilizados

solo va a ser necesario un color más.

Se quiere reducir la complejidad temporal de la siguiente función haciendo uso de programación dinámica. ¿Cuál sería la complejidad temporal resultante?

Unsigned g

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);
}
```

\*Cuadrática

Se puede reducir hasta lineal

La función no cumple con los requisitos necesarios para poder aplicar programación dinámica.

Dada la relación de recurrencia:

Dada la relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT\left(\frac{n}{a}\right) + g(n) & \text{en otro caso} \end{cases}$$

(donde  $p$  y  $a$  son enteros mayores que 1 y  $g(n) = n^k$ ), ¿qué tiene que ocurrir para que se cumpla  $T(n) \in \Theta(n^k \log_a(n))$

(donde  $p$  y  $a$  son enteros mayores que 1 y  $g(n) = n^k$ ), ¿qué tiene que ocurrir para que se cumpla  $T(n) \in \Theta(n^k \log_a(n))$ ?

$$p > a^k$$

$$p < a^k$$

$$*p = a^k.$$

Si  $f \in \Theta(g_1)$  y  $f \in \Theta(g_2)$  entonces:

$$f \in \Theta(g_1 * g_2)$$

$$f \notin \Theta(\max(g_1, g_2))$$

$$*f \in \Theta(g_1 + g_2).$$

Si  $\lim$ , ¿cuál de estas tres afirmaciones es cierta?

Si  $\lim_{n \rightarrow \infty} (f(n)/n^2) = k$ , y  $k \neq 0$ , ¿cuál de estas tres afirmaciones es cierta?

$$f(n) \in \Omega(n^3)$$

$$*f(n) \in \Theta(n^2)$$

$$f(n) \in \Theta(n^3).$$

Cogemos el algoritmo de Mergesort y en lugar de dividirlo el vector en dos partes, lo dividimos en tres. Posteriormente combinamos las soluciones parciales. ¿Cuál sería la complejidad temporal asintótica de la combinación de las soluciones parciales?

$$*\Theta(n)$$

$$\Theta(\log n)$$

Ninguna de las otras dos opciones es cierta.

Si f entonces

Si  $f \in \Theta(g_1)$  y  $f \in \Theta(g_2)$  entonces

- a  $f^2 \in \Theta(g_1 \cdot g_2)$
- b Las otras dos opciones son ambas ciertas.
- c  $f \in \Theta(\max(g_1, g_2))$

a

\*Las otras dos opciones son ambas ciertas

C.

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es cierta:

\*Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort

Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso peor del algoritmo de ordenación Quicksort

Si  $g(n) \in \Theta(1)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort.

¿Cuál de estas afirmaciones es cierta?

La ventaja de la solución de programación dinámica iterativa al problema de la mochila discreta es que nunca se realizan cálculos innecesarios

\*La memoización evita que un algoritmo recursivo ingenuo resuelva repetidamente el mismo problema

Los algoritmos iterativos de programación dinámica utilizan memoización para evitar resolver de nuevo los mismos subproblemas que se vuelven a presentar.

Dada la siguiente función:

marca la respuesta correcta:

Int exa

```
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;
}
```

La complejidad temporal en el mejor de los casos es  $\Omega(1)$

\*La complejidad temporal en el mejor de los casos es  $\Omega(n)$

La complejidad temporal exacta es  $\Theta(n \log n)$ .

La complejidad en el mejor de los casos de un algoritmo de ramificación y poda...

es siempre exponencial con el número de decisiones a tomar

suele ser polinómica con el número de alternativas por cada decisión

\*puede ser polinómica con el número de decisiones a tomar.

Indica cuál es la complejidad en el peor caso de la función replace:

```

unsigned bound(const vector<int> &v) {
 for(unsigned i = 0; i < v.size(); i++)
 if(v[i] == '0')
 return i;
 return v.size();
}

void replace(vector<int>& v, int c) {
 for(unsigned i = 0; i < bound(v); i++)
 v[i] = c;
}

```

O( $n \log n$ )

\*O( $n^2$ )

O( $n$ )

Sea  $f(n)$  la solución de la relación de recurrencia  $f(n) = 2f(n/2) + 1$ ;  $f(1) = 1$ . Indicad cuál de estas tres expresiones es cierta:

\*  $f(n) \in \Theta(n)$

$f(n) \in \Theta(n^2)$

$f(n) \in \Theta(n \log n)$

Considerad estos dos fragmentos:

$s = 0 ; \text{for } (i = 0; i < n; i++) s += i;$

y

$s=0 ; \text{for } (i=0; i < n; i++) \text{if } (a[i] \neq 0) s += i;$

y un array  $a[i]$  de números enteros. Indicad cuál de estas tres afirmaciones es cierta:

\*El coste temporal asintótico del primer programa en el caso peor es más alto que en el segundo.

El coste temporal asintótico, tanto en el caso mejor como en el caso peor, de los dos programas es el mismo.

El coste temporal asintótico del segundo programa en el caso peor es más alto que en el primero.

Indica cuál es la complejidad en función de n, donde k es una constante (no depende de n), del fragmento siguiente:

```
for(int i = k; i < n - k; i++){
 A[i] = 0;
 for(int j = i - k; j < i + k; j++)
 A[i] += B[j];
}
```

\*O(n)

O(n logn)

O(n^2)

La versión de Quicksort que utiliza como pivote la mediana del vector...

se comporta mejor cuando el vector ya está ordenado

se comporta peor cuando el vector ya está ordenado

\*el hecho de que el vector estuviera ordenado o no, no influye en la complejidad temporal de este algoritmo.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n=1 \\ \sqrt{n} + 3f(n/3) & n>1 \end{cases}$$

\* $f(n) \in \Theta(n)$

$f(n) \in \Theta(n^3)$

$f(n) \in \Theta(\sqrt{n} \log n)$

Un problema de tamaño n puede transformarse en tiempo  $O(n^2)$  en nueve de tamaño  $n/3$ , por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante.

¿cuál de estas clases de coste temporal asintótico es la más ajustada?

$O(n \log n)$

\* $O(n^2 \log n)$

$O(n^2)$

Indica cuál es la complejidad de la función siguiente:

```
unsigned sum(const mat &A) { // A es una matriz cuadrada
 unsigned d = A.n_rows();
 unsigned a = 0;
 for(unsigned i = 0; i < d; i++)
 for(unsigned j = 0; j < d; j++)
 a += A(i,j);
 return a;
}
```

O( $n^2$ )

\*O(n)

O( $n \log n$ )

Para que la complejidad de un algoritmo presente caso mejor y peor distintos...

es condición necesaria y suficiente que existan instancias distintas del problema con el mismo tamaño.

\*es condición necesaria que existan instancias distintas del problema con el mismo tamaño

es condición suficiente que existan instancias distintas del problema con el mismo tamaño.

La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.

\*las demás opciones son falsas.

siempre coincidirá con la complejidad temporal de las instancias que están en el caso base del algoritmo recursivo.

Considerad la función siguiente:

```

int M(int i, int f) {
 if (i == f)
 return i;
 else {
 e = v[M(i, (i+f)/2)];
 f = v[M((i+f)/2+1, f)];
 if (e < f)
 return e;
 else
 return f;
 }
}

```

Si la talla del problema viene dada por  $n = f - i + 1$ , ¿cuál es el coste temporal asintótico en el supuesto de que  $n$  sea una potencia de 2?

\* $O(n)$

$O(n^2)$

$O(n \log n)$

El coste temporal asintótico del fragmento

```
s=0; for(i=0;i<n;i++) for(j=i;j<n;j++) s+=i*j;
```

Y del fragmento

```
s=0; for(i=0;i<n;i++) for(j=0;j<n;j++) s+=i*i*j;
```

Son ...

\*...iguales

... el del segundo, menor que el del primero.

... el del primero, menor que el del segundo.

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. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entras las posibles según el mapa de carreteras, que este mas cercana al destino en línea recta. ¿Qué tipo de cota sería?

\*sería una cota pesimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.

ninguna de las otras dos opciones

sería una cota optimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible

Se desea encontrar el camino mas corto entre dos ciudades. Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. Como también se conocen las coordenadas geográficas de cada ciudad se quiere usar la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que este más cercana al destino según su distancia geográfica. Este algoritmo voraz, ¿serviría como cota pesimista?

\*no, ya que no asegura que se encuentre una solución factible

si, puesto que la distancia geográfica asegura que otra solución mejor no es posible.

no, ya que en algunos casos puede dar distancias menores que la optima.

El problema de cortar un tubo de longitud  $n$  en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...

no se puede resolver usando un algoritmo de vuelta atrás

\*se puede resolver mediante un algoritmo de vuelta atrás pero existe una solución asintóticamente mucho mas eficiente.

se debe resolver mediante un algoritmo de vuelta atrás, dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo.

De cual de estas tres soluciones a problemas de optimización no comporta, en el peor caso, tener que considerar  $O(n!)$  posibilidades.

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.

la solución de ramificación y poda al problema de asignación de  $n$  tareas a  $n$  trabajadores de forma de 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.

\*la solución al problema de buscar un árbol que cubre todos los vértices de un grafo de n vértices de forma que el coste mínimo (mínimum spanning tree)

Dada la suma de recurrencia

$$T(n) = \begin{cases} 1 & n=0 \\ \sum_{k=0}^{n-1} T(k) & n>0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

¿cuál de las siguientes afirmaciones es cierta?

$$T(n) \in \Theta(n^2)$$

$$T(n) \in \Theta(n!)$$

$$*T(n) \in \Theta(2^n)$$

¿Qué algoritmo es asintóticamente más rápido, el Quicksort central o el mergesort?

como su nombre indica, el Quicksort central.

son los dos igual de rápidos, ya que el coste temporal asintótico de ambos es  $O(n \log n)$

\*el mergesort es siempre más rápido o igual (salvo una constante) que el Quicksort central.

¿Cuál de estas tres expresiones es falsa?

$$2n^2 + 3n + 1 \in O(n^3)$$

$$n + n \log n \in \Omega(n)$$

$$*n + n \log n \in \Theta(n)$$

La solución recursiva ingenua a un determinado problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en...

un algoritmo del estilo de divide y vencerás.

\*un algoritmo de programación dinámica.

un algoritmo voraz.

El coste temporal del algoritmo de ordenación por selección es...

\* $O(n^2)$

$O(n)$

$O(n \log n)$ .

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

el problema de cortar un tubo de forma que se obtenga el máximo beneficio posible

el problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando las mínimas monedas.

\*el problema del viajante de comercio.

Con respecto al parámetro  $n$ , ¿Cuál es la complejidad temporal de la siguiente función?

```
Void f(unsigned n){
```

```
 If(i < 2) return;
```

```
 For(int i = 0 ; i < pow(n,2) ; i++)
```

```
 Cout “*”;
```

```
 F(n - 2);
```

```
}
```

$\Theta(n^3)$

\* $\Theta(n^2)$

$\Theta(n^2 \log n)$

En un esquema de RyP, ¿podrían coincidir los valores obtenidos por las cotas pesimistas y optimistas de un nodo cualquiera?

\*sí, en tal caso el valor obtenido por las cotas coincidiría también con el mejor valor que puede obtenerse de ese nodo.

no, en tal caso una de las cotas (o ambas) estaría mal calculada.

si, pero habría que seguir completando el nodo para saber cuál es la mejor solución que puede obtenerse de él.

Se desea resolver el problema de la potencia enésima ( $x^n$ ), asumiendo que  $n$  es par y que se utilizará la siguiente recurrencia:  $\text{pot}(x,n) = \text{pot}(x,n/2) * \text{pot}(x,n/2)$ ; ¿Qué estrategia resulta ser más eficiente en cuanto al coste temporal?

divide y vencerás

en este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.

\*un algoritmo recursivo con memoización

¿Qué tienen en común los algoritmos de ordenación Quicksort y Mergesort?

la complejidad temporal de la división en subproblemas

\*el número de llamadas recursivas que hacen en el mejor de los casos

la complejidad temporal de la combinación de las soluciones parciales.

Se pretende resolver el problema del viajante de comercio (travelling salesman problem) mediante el esquema de vuelta atrás, ¿cuál de los siguientes valores se espera que se comporte mejor para decidir si un nodo es prometedor?

\*la suma de los pesos de las  $k$  aristas restantes más cortas, donde  $k$  es el número de ciudades que quedan por visitar

el valor que se obtiene de multiplicar  $k$  por el peso de la arista más corta de entre las restantes, donde  $k$  es el número de ciudades que quedan por visitar

la suma de los pesos de las aristas que completan la solución paso a paso visitando el vértice más cercano al último visitado

Con respecto a la complejidad espacial de los algoritmos de ordenación Quicksort, Heapsort y Mergesort...

la complejidad espacial de todos ellos es lineal con el tamaño del vector a ordenar

mergesort y heapsort tienen complejidad espacial lineal con el tamaño del vector a ordenar, la de Quicksort es constante

mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante

¿Qué complejidad se obtiene a partir de la relación de recurrencia  $T(n) = 9T(n/3) + n^3$  con  $T(1) = O(1)$ ?

\* $O(n^3)$

$O(n \log n)$

$O(n^3 \log n)$

¿Cuál de las siguientes estrategias de búsqueda es más apropiada en un esquema de vuelta atrás?

explorar primero los nodos con mejor cota optimista

explorar primero los nodos con mejor valor hasta el momento en la función que se pretende optimizar

\*ninguna de las otras dos estrategias es compatible con el esquema de vuelta atrás

El algoritmo de ordenación Quicksort divide el problema en dos subproblemas. ¿Cuál es la complejidad temporal asintótica de realizar esa división?

\*ninguna de las otras dos opciones es correcta

$\Theta(\log n)$

$\Theta(n \log n)$

Queremos resolver mediante vuelta atrás el problema de las 8 reinas (colocar 8 reinas en un tablero de ajedrez de manera que no se maten mutuamente). Una buena cota optimista permitiría:

\*no es aplicable este tipo de podas a este problema

Muy probablemente, explorar menos nodos

Muy probablemente, resolver el problema de forma más rápida

Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 8T(n/8) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in \Theta(n^2)$ , ¿en cuál de estos tres casos nos podemos encontrar?

\* $g(n) = n^2$

$g(n) = n^3$

$g(n) = n$

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f (int m, int n) {
```

if (m <= n) return 1;

return m \* f(m-1,n) + n;

```
}
```

¿Qué complejidad temporal y espacial cabe esperar de la función resultante?

\* $O(m - n)$ , tanto espacial como temporal

ninguna de las otras dos opciones es correcta

$O(n - m)$ , tanto espacial como temporal

De las siguientes expresiones, o bien dos son verdaderas y una falsa, o bien dos son falsas y una es verdadera. Marca la que (en este sentido) es distinta a las otras dos

$n + n \log 2n \in \Omega(n + n \log 2n)$

\* $O(n^2) \subset O(2^{\log 2n})$

$\Omega(n^2) \subset \Omega(n)$

Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente pequeño?

que se podría podar el nodo que conduce a la solución óptima

\*que se podría explorar más nodos de los necesarios

que se podría explorar menos nodos de los necesarios

Es un algoritmo de optimización resuelto mediante ramificación y poda. ¿Podría encontrarse la solución óptima sin haber alcanzado nunca un nodo hoja?

no, los nodos hojas son los nodos completados y por lo tanto hay que visitar al menos uno de ellos para almacenarlo como la mejor solución hasta el momento

\*si, pero esto solo podría ocurrir si se hace uso de cotas pesimistas

si, esto puede ocurrir incluso si no se hace uso de cotas pesimistas

Si  $\lim f(n)/g(n) = \infty$  entonces...

... $f(n) \in O(g(n))$

... $f(n) \in \Theta(g(n))$

\* ... $f(n) \in \Omega(g(n))$

Si  $f \notin \Theta(g_1)$  y  $f \in \Theta(g_2)$  entonces siempre se cumplirá:

\* $f \in \Omega(\min(g_1, g_2))$

$f \notin O(\max(g_1, g_2))$

$f \in \Omega(g_1 + g_2)$

Decid cual de estas tres estrategias proveería la cota pesimista más ajustada al valor óptimo de la mochila discreta:

asumir que ya no se van a coger más objetos

\*completar las decisiones restantes basándose en la mejor solución voraz que pueda encontrarse para los restantes objetos y espacio disponible de la mochila

el valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es cierta:

Si  $g(n) \in \Theta(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

Si  $g(n) \in \Theta(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación mediante inserción binaria.

\*Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación mergesort

Pregunta 1

Incorrecta

Puntúa -0,50

sobre 1,00

 Marcar  
pregunta

Sea el problema "Camino de coste mínimo" con 8 movimientos: las ocho casilla colindantes. Utilizando la técnica backtracking se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para el nodo inicial calculamos una cota pesimista y otra optimista ¿Cómo debería comportarse el algoritmo en el caso de que el valor de ambas coincidan?

Seleccione una:

- a. No debería continuar.
- b. Eso no es posible: alguna de las cotas estaría mal calculada. ✗
- c. Debería continuar tratando de mejorar el valor obtenido.
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: No debería continuar.

En cuanto a la complejidad temporal asintótica de estas dos funciones, ¿cuál de las siguientes afirmaciones es correcta?

```
unsigned long pot2_1(unsigned n){
```

```
 if (n==0)
```

```
 return 1;
```

```
 return 2 * pot2_1(n-1);
```

```
}
```

```
unsigned long pot2_2(unsigned n){
```

```
 if (n==0)
```

```
 return 1;
```

```
 if (n%2==0)
```

```
 return pot2_2(n/2)*pot2_2(n/2);
```

```
 else
```

```
 return 2*pot2_2(n/2)*pot2_2(n/2);
```

```
}
```

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Ambas son equivalentes en cuanto a eficiencia
- c. La primera es más eficiente que la segunda.
- d. La segunda es más eficiente que la primera.

La respuesta correcta es: Ambas son equivalentes en cuanto a eficiencia

En función del parámetro  $n$  ¿cuál es la complejidad temporal de la siguiente función?

```
int f(int n){
 int k=0;
 for (int i = n; i > 0; i/=3)
 for (int j=i; j > 0; j-=3)
 k++;
 return k;
}
```

Seleccione una:

- a.  $\Theta(n)$
- b.  $\Theta(n \log(n))$
- c. No contesto (equivalente a no marcar nada).
- d.  $\Theta(n^2)$

La respuesta correcta es:  $\Theta(n)$

Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para tratar de acelerar la búsqueda utilizamos un algoritmo de programación dinámica aplicado al mismo problema pero restringiendo los movimientos a tres: Este, Sureste y Sur. Con este algoritmo calculamos dos valores: 1: La dificultad del mejor camino desde una casilla cualquiera  $(i,j)$  hasta el destino; 2: La dificultad del mejor camino desde el origen hasta esa casilla  $(i,j)$ . ¿con respecto al nodo que representa esa casilla, qué se puede decir de estos dos valores?

Seleccione una:

- a. Con el primero se puede componer una cota pesimista; con el segundo no se puede componer una cota optimista.
- b. No contesto (equivalente a no marcar nada).
- c. El primero es una cota pesimista; el segundo es una cota optimista. X
- d. Con el segundo se puede componer una cota pesimista; con el primero no se puede componer una cota optimista.

La respuesta correcta es: Con el primero se puede componer una cota pesimista; con el segundo no se puede componer una cota optimista.

Se pretende resolver el problema del viajante de comercio (travelling salesman problem) mediante Ramificación y poda. ¿Cuál de las siguientes acciones resulta ser mejor cota optimista para aplicarla a los nodos intermedios?

Seleccione una:

- a. Asumir que ya no quedan más vértices por visitar y cerrar el camino desde el último vértice visitado.
- b. No contesto (equivalente a no marcar nada).
- c. Obtener el árbol de recubrimiento de mínimo coste a los vértices aún no visitados.
- d. Completar el recorrido pendiente mediante un algoritmo voraz.



La respuesta correcta es: Obtener el árbol de recubrimiento de mínimo coste a los vértices aún no visitados.

Sólo una de las tres afirmaciones siguientes es cierta. Cuál?

Seleccione una:

- a. Mientras que el algoritmo de Prim va construyendo un único árbol de recubrimiento seleccionando aristas una a una, el algoritmo de Kruskal va construyendo un bosque que va uniendo añadiendo vértices hasta obtener un único árbol de recubrimiento.
- b. Mientras que el algoritmo de Prim va construyendo un único árbol de recubrimiento seleccionando vértices uno a uno, el algoritmo de Kruskal ✓ va construyendo un bosque que va uniendo añadiendo aristas, hasta obtener un único árbol de recubrimiento.
- c. Los algoritmos de Prim y de Kruskal mantienen en todo momento un único árbol de recubrimiento, que crece añadiendo aristas o vértices.
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: Mientras que el algoritmo de Prim va construyendo un único árbol de recubrimiento seleccionando vértices uno a uno, el algoritmo de Kruskal va construyendo un bosque que va uniendo añadiendo aristas, hasta obtener un único árbol de recubrimiento.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Utilizando la técnica memoización se pretende conocer la dificultad del camino más favorable. ¿Cuál es la mejor complejidad espacial y temporal que se puede conseguir?

Seleccione una:

- a.  $\Theta(n \cdot m)$ , tanto espacial como temporal.
- b. Espacial  $\Theta(n)$  y temporal  $\Theta(2^{\max(n,m)})$ .
- c. No contesto (equivalente a no marcar nada).
- d. Espacial  $\Theta(n)$  y temporal  $\Theta(n \cdot m)$ .



La respuesta correcta es:  $\Theta(n \cdot m)$ , tanto espacial como temporal.

Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para tratar de acelerar la búsqueda calculamos, para cada uno de los nodos visitados, una cota que consiste en suponer que la dificultad del camino restante es la suma de las dificultades de todas las casillas del mapa  
¿De qué tipo de cota se trata?

Seleccione una:

- a. Optimista.
- b. No contesto (equivalente a no marcar nada).
- c. Pesimista.
- d. No es cota, ni pesimista ni optimista.



La respuesta correcta es: No es cota, ni pesimista ni optimista.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Utilizando un algoritmo voraz se pretende conocer la dificultad del camino más favorable y para ello se va seleccionando una casilla escogida al azar de entre las tres posibles, hasta llegar al destino. ¿Qué podemos decir de este algoritmo?

Seleccione una:

- a. Que no es voraz.
- b. Que es voraz puesto que las decisiones no se replantean.
- c. Que aún siendo voraz existen otros criterios de selección con mejores resultados a priori.
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es: Que no es voraz.

La versión de Quicksort que utiliza como pivote la mediana del vector...

Seleccione una:

- a. ... es la versión con mejor complejidad en el mejor de los casos.
- b. No contesto (equivalente a no marcar nada).
- c. ... no presenta caso mejor y peor para instancias del mismo tamaño.
- d. ... es más eficiente si el vector ya está ordenado.

La respuesta correcta es: ... no presenta caso mejor y peor para instancias del mismo tamaño.

En un algoritmo de ramificación y poda, si la lista de nodos vivos no está clasificada de forma apropiada ...

Seleccione una:

- a. ... podría ocurrir que se exploren nodos de forma innecesaria.
- b. ... podría ocurrir que se descarten nodos factibles.
- c. ... podría ocurrir que se pade el nodo que conduce a la solución óptima.
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: ... podría ocurrir que se exploren nodos de forma innecesaria.

Con respecto a la complejidad espacial de los algoritmos de ordenación Quicksort, Heapsort y Mergesort:

Seleccione una:

- a. Mergesort y Heapsort tienen complejidad espacial lineal con el tamaño del vector a ordenar, la de Quicksort es constante.
- b. No contesto (equivalente a no marcar nada). ✗
- c. Mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante.
- d. Las complejidad espacial de todos ellos es lineal con el tamaño del vector a ordenar.

La respuesta correcta es: Mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante.

Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. A igualdad de coste temporal, ¿qué se prefiere una buena cota pesimista o una buena cota optimista?

Seleccione una:

- a. La optimista, una buena cota optimista siempre será más rentable que una buena cota pesimista. ✗
- b. No contesto (equivalente a no marcar nada).
- c. No hay distinción, la más rentable siempre será la que mejor se aproxime a la solución del nodo.
- d. La pesimista, una buena cota pesimista siempre será más rentable que una buena cota optimista.

La respuesta correcta es: No hay distinción, la más rentable siempre será la que mejor se aproxime a la solución del nodo.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur pero modificado de manera que el punto de partida es una casilla cualquiera  $(i, j)$  del mapa con nombre  $M$  ( $1 \leq i \leq n$  y  $1 \leq j \leq m$ ) y el destino la casilla  $(n, m)$ . Utilizando la técnica divide y vencerás se pretende conocer la dificultad del camino más favorable. ¿Cuál sería la recurrencia matemática más apropiada?

Seleccione una:

a.

$$\text{mcp}(n, m) = \begin{cases} \infty & \text{si } n < i \text{ o } m < j \\ M[n][m] & \text{si } n = i \text{ y } m = j \\ \min(\text{mcp}(n - 1, m - 1), \text{mcp}(n, m - 1), \text{mcp}(n - 1, m)) + M[n][m] & \text{en cualquier otro caso} \end{cases}$$

b.

$$\text{mcp}(n, m) = \begin{cases} 0 & \text{si } n < i \text{ o } m < j \\ M[i][j] & \text{si } n = i \text{ y } m = j \\ \min(\text{mcp}(n - 1, m - 1), \text{mcp}(n, m - 1), \text{mcp}(n - 1, m)) & \text{en cualquier otro caso} \end{cases}$$

c.

$$\text{mcp}(i, j) = \begin{cases} \infty & \text{si } n < i \text{ o } m < j \\ M[i][j] & \text{si } n = i \text{ y } m = j \\ \min(\text{mcp}(i - 1, j - 1), \text{mcp}(i, j - 1), \text{mcp}(i - 1, j)) + M[i][j] & \text{en cualquier otro caso} \end{cases}$$

d. No contesto (equivalente a no marcar nada).

La respuesta correcta es:  $\text{mcp}(n, m) = \begin{cases} \infty & \text{si } n < i \text{ o } m < j \\ M[n][m] & \text{si } n = i \text{ y } m = j \\ \min(\text{mcp}(n - 1, m - 1), \text{mcp}(n, m - 1), \text{mcp}(n - 1, m)) + M[n][m] & \text{en cualquier otro caso} \end{cases}$

¿Para cuál de estos problemas de optimización se conoce al menos una solución voraz óptima?

Seleccione una:

- a. El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.
- b. El problema de la mochila discreta.
- c. No contesto (equivalente a no marcar nada).
- d. 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.



La respuesta correcta es: El árbol de recubrimiento mínimo para un grafo no dirigido con pesos.

Si el coste temporal de un algoritmo es  $T(n)$ , ¿cuál de las siguientes situaciones es imposible?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $T(n) \in O(n)$  y  $T(n) \in \Theta(n)$
- c.  $T(n) \in \Theta(n)$  y  $T(n) \in \Omega(n^2)$
- d.  $T(n) \in \Omega(n)$  y  $T(n) \in \Theta(n^2)$

La respuesta correcta es:  $T(n) \in \Theta(n)$  y  $T(n) \in \Omega(n^2)$

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para tratar de acelerar la búsqueda calculamos una cota para cada uno de los nodos visitados que consiste en resolver el mismo problema pero aplicando la técnica programación dinámica con la condición de que el camino debe pasar por la casilla asociada al nodo que acabamos de visitar. ¿Qué podemos decir de esa cota?

Seleccione una:

- a. Que es una cota pesimista y optimista al mismo tiempo.
- b. Que es una cota pesimista pero no optimista.
- c. No contesto (equivalente a no marcar nada).
- d. Que en realidad no se trata de una cota, ni pesimista ni optimista.



La respuesta correcta es: Que en realidad no se trata de una cota, ni pesimista ni optimista.

Con respecto a  $n$ , ¿Cuál es la complejidad temporal del siguiente fragmento de código?

```
int a = 0;
for(int i = 0; i < n; i += 2)
 for(int j = 0; j < 10; j += 2)
 a += A[i][j];
}
```

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $\Theta(n^2)$
- c.  $\Theta(n)$
- d.  $\Theta(\log n)$

La respuesta correcta es:  $\Theta(n)$

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Se pretende listar todos los caminos posibles entre dos casillas cualesquiera ¿Qué técnica algorítmica deberíamos utilizar?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Programación dinámica.
- c. Ramificación y poda.
- d. Backtracking.



La respuesta correcta es: Backtracking.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. En cuanto a la complejidad espacial y temporal, ¿Cuál de los siguientes esquemas algorítmicos resulta más eficiente para resolverlo?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Divide y vencerás
- c. Ramificación y poda
- d. Programación dinámica



La respuesta correcta es: Programación dinámica

Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica ramificación y poda se pretende obtener la dificultad del camino más favorable desde el origen hasta la casilla destino. Para tratar de acelerar la búsqueda calculamos, para cada uno de los nodos visitados, una cota pesimista y otra optimista. ¿Cómo debería comportarse el algoritmo en el caso en el que el valor de ambas coincidan?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Aún así es necesario expandir ese nodo.
- c. El algoritmo debería terminar pues ya ha encontrado la solución. ✗
- d. Ese nodo no debería expandirse.

La respuesta correcta es: Ese nodo no debería expandirse.

Sea la recurrencia:  $T(n) = \begin{cases} 1 & n \leq 1 \\ T\left(\frac{n}{2}\right) + g(n) & n > 1 \end{cases}$  donde donde  $g(n)$  es una función polinómica. 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 diferente a las demás.

Seleccione una:

- a. Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.
- b. No contesto (equivalente a no marcar nada).
- c. Si  $g(n) \in O(n)$  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). ✗
- d. Si  $g(n) \in O(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

La respuesta correcta es: Si  $g(n) \in O(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

Por qué muchos algoritmos voraces presentan complejidades temporales en  $O(n \log n)$ ?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Porque el proceso de selección de los elementos que se incorporarán a la solución es siempre  $O(n \log n)$ .
- c. 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  $O(n \log n)$ .
- d. 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 es estrictamente inferior a  $O(n \log n)$ . ✗

La respuesta correcta es: 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  $O(n \log n)$ .

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

Seleccione una:

- a. podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.
- b. No contesto (equivalente a no marcar nada).
- c. una estrategia voraz puede ser la única alternativa.
- d. es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione.

La respuesta correcta es: una estrategia voraz puede ser la única alternativa.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Se pretende encontrar el camino más favorable con la menor complejidad temporal posible. ¿Cuál sería la más ajustada para realizar todo el proceso?

Seleccione una:

- a.  $O(n \cdot m + m)$
- b. No contesto (equivalente a no marcar nada). X
- c.  $O(n \cdot m + n + m)$
- d.  $O(n \cdot m + n)$

La respuesta correcta es:  $O(n \cdot m + n + m)$

Sea el problema "Camino de coste mínimo" con ocho movimientos (todas las casillas colindantes) pero modificado de manera que ahora se pretende encontrar el camino de máxima dificultad desde el origen hasta la casilla destino. ¿Qué cambios habría que hacer con respecto al problema sin modificar resuelto mediante ramificación y poda?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. La cota optimista se podría reutilizar sin hacerle ningún cambio, pero utilizándola como cota pesimista.
- c. Se podría intercambiar la cota pesimista por la optimista y viceversa. ✗
- d. Se podría reutilizar la cota pesimista, pero adaptándola al nuevo problema.

La respuesta correcta es: Se podría reutilizar la cota pesimista, pero adaptándola al nuevo problema.

En un mapa similar al del problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur se pretende conocer el número de caminos distintos que hay desde la casilla  $(1, 1)$  hasta una casilla cualquiera  $(i, j)$  que pertenece al mapa. El mapa tiene  $n$  filas y  $m$  columnas ¿Cuál sería la recurrencia matemática más apropiada para abordarlo mediante Divide y vencerás?

Seleccione una:

- a.  $\text{nc}(i, j) = \begin{cases} 0 & \text{si } i < 1 \text{ o } j < 1 \\ 1 & \text{si } i = 1 \text{ y } j = 1 \\ \text{nc}(i - 1, j - 1) + \text{nc}(i, j - 1) + \text{nc}(i - 1, j) & \text{en cualquier otro caso} \end{cases}$  ✓
- b.  $\text{nc}(n, m) = \begin{cases} 0 & \text{si } n < i \text{ o } m < j \\ 1 & \text{si } i = 1 \text{ y } j = 1 \\ \text{nc}(n - 1, m - 1) + \text{nc}(n, m - 1) + \text{nc}(n - 1, m) & \text{en cualquier otro caso} \end{cases}$
- c.  $\text{nc}(i, j) = \begin{cases} 0 & \text{si } n < i \text{ o } m < j \\ 1 & \text{si } i = 1 \text{ y } j = 1 \\ \text{nc}(i - 1, j - 1) + \text{nc}(i, j - 1) + \text{nc}(i - 1, j) + 1 & \text{en cualquier otro caso} \end{cases}$
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es:  $\text{nc}(i, j) = \begin{cases} 0 & \text{si } i < 1 \text{ o } j < 1 \\ 1 & \text{si } i = 1 \text{ y } j = 1 \\ \text{nc}(i - 1, j - 1) + \text{nc}(i, j - 1) + \text{nc}(i - 1, j) & \text{en cualquier otro caso} \end{cases}$

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.

Seleccione una:

a.  $\Theta(\log_2(n)) = \Theta(\log_3(n))$

b. No contesto (equivalente a no marcar nada).

c.  $\Theta(\log^2(n)) = \Theta(\log^3(n))$

d.  $\log(n^3) \notin \Theta(\log_3(n))$



La respuesta correcta es:  $\Theta(\log_2(n)) = \Theta(\log_3(n))$

:Qué tienen en común el algoritmo que obtienes el  $k$ -ésimo elemento más pequeño de un vector (estudiado en clase) y el algoritmo de ordenación.

¿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?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. La división del problema en subproblemas.
- c. El número de llamadas recursivas que se hacen.
- d. La combinación de las soluciones a los subproblemas.



La respuesta correcta es: La división del problema en subproblemas.

¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?

Seleccione una:

- a. Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.
- b. Que el algoritmo sería más lento pues se explorarían más nodos de los necesarios.
- c. No contesto (equivalente a no marcar nada).
- d. Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.

La respuesta correcta es: Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n) {
 if (n<=1)
 return 0;
 unsigned sum = desperdicio (n/2) + desperdicio (n/2);
 for (unsigned i=1; i<n-1; i++)
 for (unsigned j=1; j<=i; j++)
 for (unsigned k=1; k<=j; k++)
 sum+=i*j*k;
 return sum;
}
```

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $\Theta(n^3 \log n)$
- c.  $\Theta(n^3)$
- d.  $\Theta(2^n)$

La respuesta correcta es:  $\Theta(n^3)$

¿Cuál de las siguientes expresiones se corresponde mejor con el cálculo de la complejidad temporal asintótica en el caso peor del algoritmo que construye un Heap mínimo a partir de un vector (array) de números?

Seleccione una:

- a.  $f(n) = 2n \cdot \log_2 n - \sum_{i=0}^{\log_2 n} i \cdot 2^i \in O(n)$
- b.  $f(n) = 2n - \sum_{i=0}^{\log_2 n} i \cdot 2^i \in O(n)$
- c. Ninguna de las otras dos formulaciones es correcta.
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es:  $f(n) = 2n \cdot \log_2 n - \sum_{i=0}^{\log_2 n} i \cdot 2^i \in O(n)$

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... tiene un coste temporal asintótico exponencial.
- c. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.
- d. ... tiene la restricción de que los pesos tienen que ser enteros positivos.

La respuesta correcta es: ... tiene la restricción de que los pesos tienen que ser enteros positivos.

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor de la mochila continua correspondiente.
- c. No contesto (equivalente a no marcar nada).
- d. El valor de una mochila que contiene todos los objetos restantes aunque se pase del peso máximo permitido.

La respuesta correcta es: El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

Dada la siguiente relación de recurrencia, ¿Qué cota es verdadera?  $f(n) = \begin{cases} 1 & n = 1 \\ n + 2f(n - 1) & n > 1 \end{cases}$

Seleccione una:

- a.  $f(n) \in \Omega(2^n)$
- b.  $f(n) \in \Theta(2^n)$
- c. No contesto (equivalente a no marcar nada).
- d.  $f(n) \in O(2^n)$

La respuesta correcta es:  $f(n) \in \Omega(2^n)$

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Utilizando la técnica programación dinámica iterativa se pretende conocer la dificultad del camino más favorable. ¿Cuál es la mejor complejidad espacial y temporal que se puede conseguir?

Seleccione una:

- a. Espacial  $\Theta(n)$  y temporal  $\Theta(n \cdot m)$ . X
- b.  $\Theta(n + m)$ , tanto espacial como temporal.
- c. No contesto (equivalente a no marcar nada).
- d. Espacial  $\Theta(\min(n, m))$  y temporal  $\Theta(n \cdot m)$ .

La respuesta correcta es: Espacial  $\Theta(\min(n, m))$  y temporal  $\Theta(n \cdot m)$ .

Tenemos una función recursiva con la siguiente cabecera:

double f( const double & )

Con sólo esta información, ¿cuál podría ser una definición adecuada para el almacén?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. `vector<int> A;`
- c. Ninguna de las otras dos opciones son válidas.
- d. `vector<double> A;`

La respuesta correcta es: Ninguna de las otras dos opciones son válidas.

Si  $\lim_{n \rightarrow \infty} (g(n)/f(n))$  resulta ser cero, cuál de las siguientes expresiones puede darse?

Seleccione una:

- a. Ninguna de las otras dos opciones son ciertas.
- b. No contesto (equivalente a no marcar nada).
- c.  $f(n) \in \Omega(g(n))$  y  $g(n) \in \Omega(f(n))$
- d.  $f(n) \in \Theta(g(n))$

La respuesta correcta es: Ninguna de las otras dos opciones son ciertas.

Sea el vector  $v[8] = \{8, 6, 4, 5, 4, 3, 2, 2\}$ . ¿Cuál de las siguientes afirmaciones es falsa?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. El vector  $v$  no tiene estructura de árbol esencialmente completo y por lo tanto tampoco es un montículo
- c. El vector  $v$  tiene estructura de árbol esencialmente completo.
- d. El vector  $v$  es un montículo de máximos.

La respuesta correcta es: El vector  $v$  no tiene estructura de árbol esencialmente completo y por lo tanto tampoco es un montículo

Se quiere desarrollar un programa que compruebe si es posible que un caballo de ajedrez, mediante una secuencia de sus movimientos permitidos, recorra todas las casillas de un tablero  $N \times N$  a partir de una determinada casilla dada como entrada y sin repetir ninguna casilla. De entre las estrategias que se citan, ¿cuál sería la eficiente para resolver el problema?

Seleccione una:

- a. Algoritmo voraz.
- b. Vuelta atrás.
- c. Programación dinámica.
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es: Vuelta atrás.

¿Qué se deduce de  $f(n)$  y  $g(n)$  si se cumple  $\lim_{n \rightarrow \infty} (f(n)/g(n)) = k > 0$ ?

Seleccione una:

- a.  $f(n) \in O(g(n))$  y  $g(n) \in O(f(n))$
- b.  $f(n) \in O(g(n))$  pero  $g(n) \notin O(f(n))$
- c. No contesto (equivalente a no marcar nada).
- d.  $g(n) \in O(f(n))$  pero  $f(n) \notin O(g(n))$

La respuesta correcta es:  $f(n) \in O(g(n))$  y  $g(n) \in O(f(n))$

La solución de programación dinámica iterativa del problema de la mochila discreta...

Seleccione una:

- a. ... tiene la restricción de que los pesos tienen que ser enteros positivos.
- b. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.
- c. No contesto (equivalente a no marcar nada).
- d. ... tiene un coste temporal asintótico exponencial.

La respuesta correcta es: ... tiene la restricción de que los pesos tienen que ser enteros positivos.

**Sea el problema "Camino de coste mínimo" con ocho movimientos: todas las casillas colindantes. Utilizando la técnica de ramificación y poda se pretende encontrar el camino más favorable desde el origen hasta la casilla destino. A igualdad de coste temporal, ¿qué se prefiere una buena cota optimista o una buena cota pesimista?**

Selecione una

- a. La pesimista, una buena cota pesimista siempre será más rentable que una buena cota optimista.
- b. La optimista, una buena cota optimista siempre será más rentable que una buena cota pesimista.
- c. No contesto (equivalente a no marcar nada).
- d. No hay distinción, la más rentable siempre será la que mejor se aproxime a la solución del nodo.

La pesimista, una buena cota pesimista siempre será más rentable que una buena cota optimista.

La optimista, una buena cota optimista siempre será más rentable que una buena cota pesimista.

**No hay distinción, la más rentable siempre será la que mejor se aproxime a la solución del nodo**

¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?

Seleccione una:

- a. Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.
- b. No contesto (equivalente a no marcar nada).
- c. Que el algoritmo sería más lento pues se exploran más nodos de los necesarios.
- d. Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.

La respuesta correcta es: Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.

Tenemos una función recursiva con la siguiente cabecera:

double f( const double & )

Con sólo esta información, ¿cuál podría ser una definición adecuada para el almacen?

Seleccione una:

- a. `vector<double> A;`
- b. `vector<int> A;`
- c. No contesto (equivalente a no marcar nada).
- d. Ninguna de las otras dos opciones son válidas.

La respuesta correcta es: Ninguna de las otras dos opciones son válidas.

Sea el problema "Camino de coste mínimo" con tres movimientos: Este, Sureste y Sur. Se pretende encontrar el camino más favorable con la menor complejidad temporal posible. ¿Cuál sería la más ajustada para realizar todo el proceso?

**Sea el problema "Camino de coste mínimo" con tres movimientos Este, Sureste y Sur.**

Seleccione una:

- a.  $O(n \cdot m + n)$
- b.  $O(n \cdot m + n + m)$
- c.  $O(n \cdot m + m)$
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es:  $O(n \cdot m + n + m)$

De la 85 a la 117 no son seguras :)

Un problema de tamaño n puede transformarse en tiempo en siete de tamaño n/7 ; 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?

1. Márquez pregunta

Seleccione una:

- a.  $O(n^2)$
- b.  $O(n)$
- c.  $O(n \log n)$  ✓

De la 85 a la 117 no son seguras :)

Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a.  $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
- b.  $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$
- c.  $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$  ✓

De la 85 a la 117 no son seguras :)

4. El coste temporal asintótico del fragmento `s=0; for(i =0; i<n; i++) for(j=i; j<n; j++) s+= i*j;` y del fragmento `s=0; for(i =0; i<n; i++) for(j=i; j<n; j++) s+= i*i*j;` son...

Seleccione una:

- a. ... iguales.
- b. ... el del segundo, menor que el del primero.
- c. ... el del primero, menor que el del segundo.

Solucion: a

De la 85 a la 117 no son seguras :)

La versión de Quicksort que utiliza como pivote la mediana del vector...

Sin contestar

Puntuación 1.00

Marcar pregunta

Seleccione una:

- a. ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.
- b. ... se comporta peor cuando el vector ya está ordenado.
- c. ... se comporta mejor cuando el vector ya está ordenado.

Solución: El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

De la 85 a la 117 no son seguras :)

La complejidad temporal en el mejor de los casos...

Sin contestar  
Puntúa como 1.00  
 Marcar pregunta

Seleccione una:

a. ... es una función de la talla que tiene que estar definida para todos los posibles valores de ésta.

b. Las demás opciones son verdaderas.

c. ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.

Solucion: A

De la 85 a la 117 no son seguras :)

Pertenece  $3n^2 + 3$  a  $O(n^3)$ ?

a) No

**b) Sí**

c) Solo para  $c = 1$  y  $n_0 = 5$

De la 85 a la 117 no son seguras :)

Indica cuál es la complejidad en función de n, donde k es una constante(no depende de n) del fragmento siguiente:

Puntaje: 0.00  
Punto:   
Variable:   
seguro

```
for (int i = 0 ; i < n - k ; i++)
 A[i] = B[i]
 for (int j = i + k ; j < i + B[j] ; j++)
 A[i] += B[j];
 }
```

Selecciona una:  
 a  $O(n \log n)$   
 b  $O(n)$   
 c  $O(n^2)$

Solucion:  $O(n)$

De la 85 a la 117 no son seguras :)

Indica cuál es la complejidad, en función de n, del fragmento siguiente:

Puntúa como 1,00

▼ Marcar pregunta

```
k=n/4;
for(int i = k; i < n - k; i++){
 A[i] = 0;
 for(int j = i - k; j < i + k; j++)
 A[i] += B[j];
}
```

Seleccione una:

- a.  $O(n \log n)$
- b.  $O(n)$
- c.  $O(n^2)$

Solucion:  $O(n^2)$

De la 85 a la 117 no son seguras :)

Un problema de tamaño N puede transformarse en tiempo  $O(N^2)$  en nueve de tamaño  $N/3$ ; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante. ¿cuál de estas clases de coste temporal asintótico es la más ajustada?

pregunta

Seleccione una:

- a.  $O(n^2 \log n)$  ✓
- b.  $O(n \log n)$
- c.  $O(n^2)$

De la 85 a la 117 no son seguras :)

¿Cuál es la complejidad temporal de la siguiente función recursiva?

Corrección  
Puntuación 1.00  
Términos y condiciones

```
unsigned desperdicio (unsigned n) {
 if (n<=1)
 return 0;
 unsigned sum = desperdicio (n/2) + desperdicio (n/2) + desperdicio (n/2);
 for (unsigned i=1; i<n-1; i++)
 for (unsigned j=1; j<=i; j++)
 sum+=i*j;
 return sum;
}
```

Selección una:

a.  $\Theta(n^2)$  ✓  
 b.  $\Theta(n^2 \log n)$   
 c.  $\Theta(3^n)$

$O(n^2)$

De la 85 a la 117 no son seguras :)

Indicad cual es la complejidad de la función siguiente:

```
unsigned sumi(const mat &A) { // A es una matriz cuadrada
 unsigned d = A.n_rows();
 unsigned s = 0;
 for(unsigned i = 0; i < d; i++)
 for(unsigned j = 0; j < d; j++)
 s += A(i,j);
 return s;
}
```

Seleccione una:

- a  $O(n^2)$
- b  $O(n)$  ✓
- c  $O(n \log n)$

De la 85 a la 117 no son seguras :)

Indica cuál es la complejidad, en función de  $n$ , del siguiente fragmento de código:  $s=0; \text{for}(i=0;i<n;i++) \text{for}(j=i;j<n;j++) s+=i*j;$

Marcar pregunta

Seleccione una:

a.  $\Theta(n)$

b.  $\Theta(n^2)$  ✓

c.  $O(n^2)$  pero no  $\Omega(n^2)$ .

De la 85 a la 117 no son seguras :)

Indica cuál de estas tres expresiones es falsa:

Seleccione una:

- a.  $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$  ✓
- b.  $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
- c.  $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$

De la 85 a la 117 no son seguras :)

Sea  $f(n)$  la solución de la relación de recurrencia  $f(n)=2f(n/2)+1$   $f(1)=1$ . Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a.  $f(n) \in \Theta(n)$  ✓
- b.  $f(n) \in \Theta(n \log(n))$
- c.  $f(n) \in \Theta(n^2)$

De la 85 a la 117 no son seguras :)

Un problema de tamaño N puede transformarse en tiempo  $O(N^3)$  en ocho de tamaño  $N/3$ ; por otro lado, la solución al problema cuando la talla es 1 requiere un tiempo constante. ¿cuál de estas clases de coste temporal asintótico es la más ajustada?

Seleccione una:

- a.  $O(n^3)$
- b.  $O(n^2 \log n)$
- c.  $O(n^3 \log n)$  ✓

De la 85 a la 117 no son seguras :)

Dada la siguiente cota de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n=1 \\ n+3f(n/3) & n>1 \end{cases}$$

Seleccione una:

- a.  $f(n) \in \Theta(n \log n)$  ✓
- b.  $f(n) \in \Theta(n)$
- c.  $f(n) \in \Theta(n^3)$

De la 85 a la 117 no son seguras :)

¿Cuál es la complejidad temporal de la siguiente función recursiva?

$O(n^3)$

Pregunta 5

Completada

Puntúa como 1,00

Marcar pregunta

¿Cuál es la complejidad temporal de la siguiente función recursiva?

```
unsigned desperdicio (unsigned n) {
 if (n<=1)
 return 0;
 unsigned sum = desperdicio (n/2) + desperdicio (n/2);
 for (unsigned i=1; i<n-1; i++)
 for (unsigned j=1; j<i; j++)
 for (unsigned k=1; k<j; k++)
 sum+=i*j*k;
 return sum;
}
```

Seleccione una:

a.  $\Theta(n^3)$  ✓

b.  $\Theta(n^3 \log n)$

c.  $\Theta(2^n)$

Seleccione una:

- a. El coste temporal asintótico del primer programa en el caso peor es más alto que en el segundo.
- b. El coste temporal asintótico del segundo programa en el caso peor es más alto que en el primero.
- c. El coste temporal asintótico, tanto en el caso mejor como en el caso peor, de los dos programas es el mismo. ✓

De la 85

Considerad estos dos fragmentos:  $s=0; \text{for}(i=0;i<n;i++) s+=i;$  y  $s=0; \text{for}(i=0;i<n;i++) \text{if}(a[i]!=0)s +=i;$  ; y un array  $a[i]$  de números enteros. Indicad cuál de estas tres afirmaciones es cierta:

El coste temporal asintótico, tanto en el caso mejor como en el caso peor de los dos programas es el mismo.

De la 85 a la 117 no son seguras :)

Los algoritmos de ordenación Quicksort y Mergesort tienen en común...

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- 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.

... que aplican la estrategia de divide y vencerás.

De la 85 a la 117 no son seguras :)

La complejidad temporal en el mejor de los casos de un algoritmo recursivo:

Seleccione una:

- a. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.
- b. Las demás opciones son falsas. ✓
- c. ... siempre coincidirá con la complejidad temporal de las instancias que están en el caso base del algoritmo recursivo.

De la 85 a la 117 no son seguras :)

Indica cuál es la complejidad, en función de n, del fragmento siguiente:

Correcta

Puntuá como 1,00

▼ Marcar  
pregunta

```
int a = 0;
for(int i = 0; i < n; i++)
 for(int j = i; j > 0; j /= 2)
 a += A[i][j];
```

Seleccione una:

- \* a.  $O(n \log n)$  ✓
- b.  $O(n^2)$
- c.  $O(n)$

De la 85 a la 117 no son seguras :)

Indica cuál de estas tres expresiones es falsa:



Seleccione una:

- a.  $\Theta(n) \subseteq O(n)$
  - b.  $\Theta(n/2) = \Theta(n)$
  - c.  $\Theta(n) \subseteq \Theta(n^2)$  ✓
-

De la 85 a la 117 no son seguras :)

¿Cuál de estas tres expresiones es falsa?

$$3n^2 + 1 \in O(n^3)$$

$$n + n\log(n) \in \Theta(n)$$
 ✓

$$n + n\log(n) \in \Omega(n)$$

De la 85 a la

Seleccione una:

a.  $O(n)$  ✓

b.  $O(n^2)$ .

c.  $O(n \log(n))$

Considerad la función siguiente:

Si no contestar  
Puntúa como 1.00  
 Marcar pregunta

```
int H(int i, int f) {
 if (i == f)
 return i;
 else {
 e = v[H(i, (i+f)/2)];
 f = v[H((i+f)/2+1, f)];
 if (e < f)
 return e;
 else
 return f;
 }
}
```

Si la tamañía del problema viene dada por  $\gamma_1 := j - i + 1$ , ¿cuál es el costo temporal asintótico en el supuesto de que  $\gamma_1$  sea una potencia de 2?

De la 85 a la 117 no son seguras :)

Dada la siguiente cota de recurrencia, ¿Qué cota es verdadera?

Correcta

Puntúa como 1,00

Marcar pregunta

$$f(n) = \begin{cases} 1 & n=1 \\ n^2 + 3f(n/3) & n>1 \end{cases}$$

Seleccione una:

- a.  $f(n) \in \Theta(n)$
- b.  $f(n) \in \Theta(n^2)$  ✓
- c.  $f(n) \in \Theta(n^2 \log n)$

De la 85 a la 117 no son seguras :)

Sea  $f(n)$  la solución de la relación de recurrencia  $f(n)=2f(n-1)+1$   $f(1)=1$ . Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a  $f(n) \in \Theta(2^n)$
- b  $f(n) \in \Theta(n)$  X
- c  $f(n) \in \Theta(n^2)$

Solución:  $2^n$

De la 85 a la 117 no son seguras :)

Pregunta 12

Sin responder aún

Puntaje correcto 1,00

Marcar  
incorrecta

De las siguientes expresiones, o bien dos son verdaderas y una falsa, o bien dos son falsas y una verdadera. Marca la que es verdadera y da clic en intentar si es otra.

Seleccione una:

- a  $O(\log(n^2)) = O(\log(n^4))$
- b  $O(\log_2(n)) = O(\log_3(n))$
- c  $O(\log^2(n)) = O(\log^3(n))$

De la 85 a la 117 no son seguras :)

Problema 2  
Para responder más  
Puntaje como 1.00  
V<sup>Y</sup> Marcar  
problema

Estado de siguientes fragmentos de código

Complejidad: <https://www.wolframalpha.com>

Indica cuál de las siguientes expresiones, es la que mejor refleja su complejidad temporal.

Selección: una

A.  $\sum_{i=0}^n \log i$

B.  $\sum_{i=0}^n \log n$

C. Ninguna de las otras dos opciones son correctas

De la 85 a la 117 no son seguras :)

Indica cuál es la complejidad en el peor caso de la función replace:

```
unsigned bound(const vector<int> &v) {
 for(unsigned i = 0; i < v.size(); i++)
 if(v[i] == 0)
 return i;
 return v.size();
}

void replace(vector<int>& v, int c) {
 for(unsigned i = 0; i < bound(v); i++)
 v[i] = c;
}
```

$O(n^2)$

De la 85 a la 117 no son seguras :)

Un programa con dos bucles anidados uno dentro de otro. El primero hace n iteraciones aproximadamente y el segundo la mitad, tarda un tiempo

Seleccione una:

- a.  $O(n \log n)$
- b.  $O(n^2)$  ✓
- c.  $O(n\sqrt{n})$

De la 85 a la 117 no son seguras :)

Sea  $f(n)$  la solución de la relación de recurrencia  $f(n)=2f(n-1)+n$   $f(1)=1$ . Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a.  $f(n) \in \Theta(n^2)$
- b.  $f(n) \in \Theta(n)$
- c.  $f(n) \in \Theta(n \log(n))$  ✓

De l

Para que la complejidad de un algoritmo presente caso mejor y peor distintos ...

Seleccione una:

- a. ... es condición necesaria y suficiente que existan instancias distintas del problema con el mismo tamaño.
- b. ... es condición necesaria que existan instancias distintas del problema con el mismo tamaño. ✓
- c. ... es condición suficiente que existan instancias distintas del problema con el mismo tamaño.

Para que la complejidad de un algoritmo presente caso mejor y peor distintos ...

**... es condición necesaria que existan instancias distintas del problema con el mismo tamaño.**

De la 85 a la 117 no son seguras :)

Indicad cuál de estas tres expresiones es cierta:

Seleccione una:

- a.  $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
- b.  $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$
- c.  $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$  ✓

De la 85 a la 117 no son seguras :)

considerar la función siguiente:

Si la talla del problema viene dada por  $n = f-i+1$ . ¿Cuál es el coste temporal asintótico en el supuesto de que  $n$  sea una potencia de 2?

$O(n)$

Considerad la función siguiente:

```
int M(int i, int f) {
 if (i == f)
 return i;
 else {
 e = v[m(i, (i+f)/2)];
 f = v[m((i+f)/2 +1, f)];
 if (e < f)
 return e;
 else
 return f;
 }
}
```

Si la talla del problema viene dada por  $n = f-i+1$ . ¿Cuál es el coste temporal asintótico en el supuesto de que  $n$  sea una potencia de 2?

Selección una:

- a  $O(n)$
- b  $O(n^2)$
- c  $O(n \log(n))$

De la 85 a la 117 no son seguras :)

La versión de Quicksort que utiliza como pivote el elemento del vector que ocupa la primera posición...

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado.
- b. ... se comporta peor cuando el vector ya está ordenado. ✓
- c. ... El hecho de que el vector estuviera previamente ordenado o no, no influye en la complejidad temporal de este algoritmo.

De la 85 a la 117 no son seguras :)

La versión de Quicksort que utiliza como pivote el elemento del vector que ocupa la posición central

Seleccione una:

- a. ... se comporta mejor cuando el vector ya está ordenado. ✓
- b. ... se comporta peor cuando el vector ya está ordenado.
- c. ... no presenta casos mejor y peor distintos para instancias del mismo tamaño.

De la 85 a la 117 no son seguras :)

¿Cuál es el objetivo de la etapa de análisis en el Diseño y Análisis de un Algoritmo?:

- a. Determinar el lenguaje y herramientas disponibles para su desarrollo.
- b. Estimar los recursos que consumirá el algoritmo una vez implementado.**
- c. Estimar la potencia y características del equipo informático necesarios para el correcto funcionamiento del algoritmo.

De la 85 a la 117 no son seguras :)

¿Cuál de las siguientes jerarquías de complejidades es la correcta?

a.  $O(1) \subset O(\lg n) \subset O(\lg \lg n) \subset \dots$

b.  $\dots \subset (n!) \subset O(2^n) \subset O(n^n)$

**c.  $\dots \subset (2^n) \subset O(n!) \subset O(n^n)$**

De la 85 a la 117 no son seguras :)

¿Cuál de los siguientes algoritmos de ordenación tiene menor complejidad?

- a. Burbuja
- b. Inserción directa
- c. Mergesort**

De la 85 a la 117 no son seguras :)

¿El tiempo de ejecución de un algoritmo depende de la talla del problema?

- a. Sí, siempre
- b. No, nunca
- c. No necesariamente**

De la 85 a la 117 no son seguras :)

Ordena de menor a mayor las siguientes complejidades

1.  $O(1)$
2.  $O(n^2)$
3.  $O(n \lg n)$
4.  $O(n!)$

a. 3,1,2 y 4

b. 1,3,2 y 4

c. 1,3,4 y 2

De la 85 a la 117 no son seguras :)

El estudio de la complejidad resulta realmente interesante para tamaños grandes de problema por varios motivos:

- a. Las diferencias reales en tiempo de compilación de algoritmos con diferente coste para tamaños pequeños del problema no suelen ser muy significativas.
- b. Las diferencias reales en tiempo de ejecución de algoritmos con diferente coste para tamaños grandes del problema no suelen ser muy significativas.
- c. Ninguna de las anteriores.**

De la 85 a la 117 no son seguras :)

¿Por qué se emplean funciones de coste para expresar el coste de una algoritmo?

- a. Para poder expresar el coste de los algoritmos con mayor exactitud
- b. Para que la expresión del coste del algoritmo sea válida para cualquier entrada al mismo**
- c. Para poder expresar el coste de un algoritmo mediante una expresión matemática

De la 85 a la 117 no son seguras :)

El caso base de una ecuación de recurrencia asociada a la complejidad temporal de un algoritmo expresa:

- a. El coste de dicho algoritmo en el mejor de los casos.
- b. El coste de dicho algoritmo en el peor de los casos.
- c. Ninguna de las anteriores**

De la 85 a la 117 no son seguras :)

La complejidad de la función TB es:

```
función TB (A: vector[]; iz , de : N) : N
 var n,i:N;
 n=iz-de+1
 opción
 (n < 1) : devuelve (0) ;
 (n = 1) : devuelve (1) ;
 (n > 1) : si (A[iz] = A[de]) entonces
 devuelve (TB(A, iz + 1, de - 1) + 1);
 sino
 devuelve (TB(A, iz + 1, de - 1)) ;
 finsi ;
 fopcion
fin
```

- a.  $\Theta(n)$
- b.  $\Theta(n \lg n)$
- c.  $\Theta(n^2 \lg n)$

De la 85 a la 117 no son seguras :)

Dado el polinomio  $f(n) = a_m n^m + a_{m-1} n^{m-1} + \dots + a_0$ , con  $a_m \in \mathbb{R}^+$  entonces  $f$  pertenece al orden:

- a.  $O(n^m)$ .
- b.  $\Omega(n^m)$ .
- c. **La dos respuestas anteriores son correctas.**

De la 85 a la 117 no son seguras :)

Si  $f_1(n) \in \%(\text{g1}(n))$  y  $f_2(n) \in O(\text{g2}(n))$  entonces:

a.  $f_1(n)+f_2(n) \in O(\max(\text{g1}(n), \text{g2}(n)))$

b.  $f_1(n)+f_2(n) \in O(\text{g1}(n)+\text{g2}(n))$

**c. Ambas son correctas**

De la 85 a la 117 no son seguras :)

Un algoritmo cuya talla es  $n$  y que tarda  $40^n$  segundos en resolver cualquier instancia tiene una complejidad temporal:

- a.  $O(n^n)$
- b.  $O(4^n)$**
- c. Ninguna de las anteriores

De la 85 a la 117 no son seguras :)

Si dos algoritmos tienen la misma complejidad asintótica:

- a. **No necesitan exactamente el mismo tiempo para su ejecución.**
- b. Necesitan exactamente el mismo tiempo para su ejecución.
- c. Ninguna de las anteriores

De la 85 a la 117 no son seguras :)

Los algoritmos directos de ordenación, respecto de los indirectos:

- a. **Presentan una mayor complejidad temporal y sus tiempos de ejecución absolutos son mayores.**
- b. Presentan una menor complejidad temporal y sus tiempos de ejecución absolutos son menores.
- c. Presentan una mayor complejidad temporal si bien sus tiempos de ejecución absolutos son menores.

De la 85 a la 117 no son seguras :)

La talla o tamaño de un problema depende de:

- a. Conjunto de valores asociados a la entrada y salida del problema.
- b. Conjunto de valores asociados a la salida del problema.
- c. Conjunto de valores asociados a la entrada del problema.**

De la 85 a la 117 no son seguras :)

En un algoritmo recursivo, la forma de dividir el problema en subproblemas:

- a. Influye en la complejidad espacial del mismo.
- b. Influye en su complejidad temporal.**
- c. No influye en ninguna de sus complejidades.

De la 85 a la 117 no son seguras :)

$f(n) = 5n + 3m \cdot n + 11$  entonces  $f(n)$  pertenece a:

- a.  $O(n \cdot m)$ .
- b.  $O(n^m)$ .
- c. Las dos son correctas

De la 85 a la 117 no son seguras :)

El sumatorio, desde  $i=1$  hasta  $n$ , de  $i^k$  pertenece a:

- a.  $O(n^{k+1})$
  - b.  $O(n^k)$
  - c. Ninguna de las anteriores
-

De la 85 a la 117 no son seguras :)

La complejidad de la función A2 es:

Funcion A2 (n, a; entero):entero;

Var r: entero; fvar

    si ( $a^2 > n$ ) devuelve 0

    sino

        r := A2(n, 2a);

        opción

            n <  $a^2$ :      devuelve r;

            n  $\geq a^2$ :      devuelve r + a;

        fopción

    fsi

fin

a.  $O(\sqrt{n} \cdot a)$

b.  $O(\sqrt{n} / a)$

c.  $O(n / \sqrt{a})$

De la 85 a la 117 no son seguras :)

Cual de las siguientes definiciones es cierta:

- a. Las cotas de complejidad se emplean cuando para una misma talla se obtienen diferentes complejidades dependiendo de la entrada al problema.**
- b. Las cotas de complejidad se emplean cuando para diferentes tallas se obtienen diferentes complejidades dependiendo de la entrada al problema.
- c. Ninguna de las anteriores

De la 85 a la 117 no son seguras :)

Cuando para distintas instancias de problema con el mismo tamaño no obtenemos el mismo resultado:

- a. No es posible calcular la complejidad a priori y debemos ejecutar el programa varias veces con la misma talla y obtener el tiempo medio para hallar la complejidad media.
- b. No se puede aplicar la técnica de paso de programa, ya que esta técnica es para calcular la complejidad a priori.
- c. Calculamos el máximo y mínimo coste que nos puede dar el algoritmo.**

De la 85 a la 117 no son seguras :)

$f(n) = 5n+5$  ¿  $f(n)$  pertenece a  $O(n)$ ?

- a. Si. El valor de  $c$  es 5 y el valor mínimo de  $n_0$  es de 3
- b. Si. El valor de  $c$  es 9 y el valor mínimo de  $n_0$  es de 1
- c. Si. El valor de  $c$  es 6 y el valor mínimo de  $n_0$  es de 5**

De la 85 a la 117 no son seguras :)

$f(n) = 10n+7$  ;  $f(n)$  pertenece a  $O(n^2)$ ?

- a. Si. Para  $c = 1$  y a partir de un valor de  $n_0 = 10$ .
- b. Sí Para cualquier valor de  $c$  positivo siempre existe un  $n_0$  a partir del que se cumple.**
- c. No.

De la 85 a la 117 no son seguras :)

Si  $f(n) \in \Omega(g(n))$  entonces:

- a.  $\exists c, n_0 \in \mathbb{R}^+: f(n) \geq c \cdot g(n) \quad \forall n \geq n_0$
  - b.  $\exists c, n_0 \in \mathbb{R}^+: f(n) \geq c \cdot g(n) \quad \forall n$
  - c.  $\exists c, n_0 \in \mathbb{R}^+: f(n) \leq c \cdot g(n) \quad \forall n \geq n_0$
-

De la 85 a la 117 no son seguras :)

El coste asociado a la siguiente ecuación de recurrencia es:

$$f(n) = \begin{cases} 1 & n \leq 1 \\ n + f(n/2) + f(n/2) & n > 1 \end{cases}$$

- a.  $\Theta(n \lg n^2)$
  - b.  $\Theta(n^2 \lg n)$
  - c.  $\Theta(n \lg n)$
-

De la 85 a la 117 no son seguras :)

El estudio de la complejidad tamaños grandes de problema por varios motivos:

- a. Las diferencias reales en tiempo de compilación de los algoritmos con diferente coste para tamaños pequeños del problema no suelen ser muy significativas.
- b. Las diferencias reales en tiempo de ejecución de algoritmos con diferente coste para tamaños grandes del problema no suelen ser muy significativas.
- c. Ninguna de las anteriores.

De la 85 a la 117 no son seguras :)

¿Por qué se emplean funciones de coste para expresar el coste de un algoritmo?

- a. Para poder expresar el coste de los algoritmos con mayor exactitud.
- b. Para que la expresión del coste del algoritmo sea cualquier entrada del mismo.
- c. Para poder expresar el coste de un algoritmo mediante una expresión matemática.

De la 85 a la 117 no son seguras :)

Si la complejidad temporal de la función F2 complejidad temporal del siguiente algoritmo es:

```
funcion suma(a:entero)
 m := 5*a
 para i:= 1 hasta m
 para j := 1 hasta m
 F2(a)
 fpara
 fpara
ffuncion
```

- a.  $a^4$
- b.  $a^2$
- c.  $a^3$

De la 85 a la 117 no son seguras :)

El coste asintótico de la siguiente función es:

- a.  $\Theta(n)$ .
- b.  $\Theta(n^2)$ .
- c.  $\Theta(\log(n))$ .

```
funcion factorial(int n){
 if(n == 0) return n;
 else return n*factorial(n-1);
}
```

De la 85 a la 117 no son seguras :)

Indica cual es la complejidad, en función de n, del fragmento siguiente:

a. **O(n log n)**

```
int a = 0;
```

b.  $O(n^2)$

```
for(int i = 0; i < n; i++)
 for(int j = i; j > 0; j /= 2)
 a += A[i][j];
```

c.  $O(n)$

De la 85 a la 117 no son seguras :)

Un algoritmo recursivo basado en el esquema de divide y vencerás:

- a. Será más eficiente cuantos más equitativa sea la división en subproblemas.
- b. Nunca tendrá una complejidad exponencial.
- c. Las demás opciones son verdaderas.

De la 85 a la 117 no son seguras :)

La complejidad temporal en el mejor de los casos... Seleccione una:

- a. ... es el tiempo que tarda el algoritmo en resolver la talla más pequeña que se le puede presentar.
- b. Las demás opciones son verdaderas.
- c. ... es una función de la talla que tiene que estar definida para todos los posibles valores de ésta.**

De la 85 a la 117 no son seguras :)

Sobre la complejidad temporal de la siguiente función: Seleccione una:

- a. Ninguna de las otras dos alternativas es cierta.**
- b. Las complejidades en los casos mejor y peor son distintas.
- c. El mejor de los casos se da cuando  $n \leq 1$  y en tal caso la complejidad es constante.

```
unsigned desperdicio (unsigned n) {
 if (n<=1)
 return 0;
 unsigned sum = desperdicio (n/2) + desperdicio (n/2) + desperdicio
 (n/2);
 for (unsigned i=1; i<n-1; i++)
 for (unsigned j=1; j<=i; j++)
 for (unsigned k=1; k<=j; k++)
 sum+=i*j*k;
 return sum;
}
```

De la 85 a la 117 no son seguras :)

Con respecto al esquema Divide y vencerás, ¿es cierta la siguiente afirmación? Si la talla se reparte equitativamente entre los subproblemas, entonces la complejidad temporal resultante es una función logarítmica. Seleccione una:

- a. No, puesto que también hay que añadir el coste de la división en subproblemas y la posterior combinación.
- b. No tiene por qué, la complejidad temporal no depende únicamente del tamaño resultante de los subproblemas.**
- c. Sí, siempre, en Divide y Vencerás la complejidad temporal depende únicamente del tamaño de los subproblemas.

De la 85 a la 117 no son seguras :)

¿Qué cota se deduce de la relación de recurrencia?

$$f(n) = \begin{cases} 1 & n=1 \\ n+4f(n/2) & n>1 \end{cases}$$

Seleccione una:

- a.  $f(n) \in \Theta(n^2)$
- b.  $f(n) \in \Theta(n)$
- c.  $f(n) \in \Theta(n \log n)$

De la 85 a la 117 no son seguras :)

¿Cuál de estas tres expresiones es falsa?

Seleccione una:

- a.  $2n^3 - 10n^2 + 1 \in O(n^3)$
- b.  $n + n\sqrt{n} \in \Omega(n)$
- c.  $n + n\sqrt{n} \in \Theta(n)$

La respuesta correcta es:  $n + n\sqrt{n} \in \Theta(n)$

De la 85 a la 117 no son seguras :)

Sea  $f(n) = n \log(n) + n$

Seleccione una:

- a. ...  $f(n) \in \Omega(n \log(n))$
- b. ...  $f(n) \in O(n \log(n))$
- c. Las otras dos opciones son ciertas

La respuesta correcta es: Las otras dos opciones son ciertas

De la 85 a la 117 no son seguras :)

Si  $f_1(n) \in O(g_1(n))$  y  $f_2(n) \in O(g_2(n))$  entonces:

Marcar pregunta

Seleccione una:

- a. Las otras dos alternativas son ciertas.
- b.  $f_1(n)+f_2(n) \in O(\max(g_1(n), g_2(n)))$
- c.  $f_1(n)+f_2(n) \in O(g_1(n)+g_2(n))$

La respuesta correcta es: Las otras dos alternativas son ciertas.

Sin contestar

Puntuación 1,00

Marcar  
pregunta

```
int ejemplo (vector < int > & v) {
 int n=v.size();
 int j,i=2;
 int sum=0;
 while (n>0 && i<n) {
 j=i;
 while (v[j] != v[i]){
 sum+=v[j];
 j=j/2;
 }
 }
}
```

¿Cuál es la complejidad temporal de la siguiente función?

---

<https://moodle2013-14.ua.es/moodle/mod/quiz/review>

```
i++;
}
return sum;
```

Seleccione una:

- a.  $\Theta(n \log n)$
- b.  $\Theta(n^2)$
- c.  $\Omega(n)$

La respuesta correcta es:  $\Theta(n \log n)$

De la 85 a la 117 no son seguras :)

En cuanto a la complejidad temporal de la siguiente función:

Sin contexto  
Puntaje como 1,00  
V. Hacer  
proyecto

```
int ejemplo (vector < int > v) {
 int n=v.size();
 int j,i=2;
 int sum=0;
 while (n>0 && i<n){
 j=i;
 while (v[j] < v[i]){
 sum+=v[j];
 j=j/2;
 }
 i++;
 }
 return sum;
}
```

Seleccione una:

- a. Las complejidades en el mejor y en el peor de los casos no coinciden.
- b. El mejor de los casos se da cuando  $n = 0$ , su complejidad es constante.
- c. Esta función no presenta casos mejor y peor puesto que sólo puede haber una instancia para cada una de las posibles tallas

La respuesta correcta es: Las complejidades en el mejor y en el peor de los casos no coinciden.

De la 85 a la 117 no son seguras :)

Indica cuál es la complejidad, en función de n, del fragmento siguiente:

```
for(int i = n; i > 0; i /= 2)
 for(int j = n; j > 0; j /= 2)
 a += A[i][j];
```

Seleccione una:

- a.  $O(\log^2(n))$
- b.  $O(n \log(n))$
- c.  $O(n^2)$

De la 85 a la 117 no son seguras :)

Indica cuál es la complejidad, en función de n, del fragmento siguiente:

```
s = 0
for(int i = 0; i < n*n; i++)
 s += A[i + j / n];
```

Selecione una:

---

<https://moodle2013-14.us.es/moodle/mod/>

79

- a.  $O(n^2)$
- b.  $O(n \log(n))$
- c.  $O(n)$

La respuesta correcta es:  $O(n^2)$

De la 85 a la 117 no son seguras :)

La versión de Quicksort que utiliza como pivote la mediana del vector... Seleccione una:

- a. ... no presenta caso mejor y peor distintos para instancias del mismo tamaño.**
- b. ... es más eficiente si el vector ya está ordenado.
- c. ... es la versión con mejor complejidad en el mejor de los casos.

De la 85 a la 117 no son seguras :)

El siguiente fragmento del algoritmo de ordenación Quicksort reorganiza los elementos del vector para obtener una subsecuencia de elementos menores que el pivote y otra de mayores. Su complejidad temporal, con respecto al tamaño del vector v, que está delimitado por los valores pi y pf, es...

```
x = v[pi];
i = pi+1;
j = pf;
do {
 while(i <= pf && v[i] < x) i++;
 while(v[j] > x) j--;
 if(i <= j) {
 swap(v[i],v[j]);
 i++;
 j--;
 }
} while(i < j);
swap(v[pi],v[j]);
```

Nota: La función swap se realiza en tiempo constante.

Seleccione una:

- a. ... lineal en cualquier caso,
- b. ... cuadrática en el peor de los casos.
- c. ... lineal en el caso peor y constante en el caso mejor.

De la 85 a la 117 no son seguras :)

Dada la siguiente cota de recurrencia, ¿Qué cota es verdadera?

$$f(n) = \begin{cases} 1 & n=1 \\ n+2f(n-1) & n>1 \end{cases}$$

Seleccione una:

- a.  $f(n) \in \Omega(2^n)$
- b.  $f(n) \in \Theta(n^2)$
- c.  $f(n) \in \Theta(2^n)$

De la 85 a la 117 no son seguras :)

¿Cual es la solución a la siguiente relación de recurrencia?

Sin contestar  
 Puntuación 1,00  
 Haré pregunta

$$f(n) = \begin{cases} \Theta(1) & n=0 \\ \Theta(1)+f(n/3) & n>0 \end{cases}$$

Seleccione una:

- a.  $f(n) \in \Theta(\log(n))$ .
- b.  $f(n) \in \Theta(n/3)$
- c. Ninguna de las otras dos es cierta.

La respuesta correcta es:  $f(n) \in \Theta(\log(n))$

De la 85 a la 117 no son seguras :)

Recorrer solo uno de los triángulos (el superior o el inferior de una matriz) de una matriz  $n \times n$  tiene una complejidad asintótica, con respecto a  $n$  que está en...

Recorrer sólo uno de los triángulos (el superior o el inferior) de una matriz  $n \times n$  tiene una complejidad asintótica, con respecto a  $n$ , que está en ...

Seleccione una:

a.  $O(n\sqrt{n})$

b.  $O(n \log n)$

Respuesta: C

c.  $\Omega(n^2)$

De la 85 a la 117 no son seguras :)

Las siguientes funciones calculan el valor de la potencia n-ésima de dos.

Las siguientes funciones calculan el valor de la potencia n-ésima de dos. ¿Cuál es más eficiente en cuanto a coste temporal?

```
unsigned long pot2_1(unsigned n) {
 if (n==0) return 1;
 else return pot2_1(n/2) * pot2_1(n/2);
 else return 2 * pot2_1(n/2) + pot2_1(n/2);
```

```
unsigned long pot2_2(unsigned n) {
 if (n==0) return 1;
 return 2 * pot2_2(n-1);
```

Selcción correcta:

- a. La primera, pot2\_1(), es más eficiente que la otra.
- b. La segunda, pot2\_2(), es más eficiente que la otra.
- c. No cambia (equivalente a no hacer nada).
- d. Las dos funciones son equivalentes en cuanto a coste temporal.

De la 85 a la 117 no son seguras :)

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo. ¿Cuál es una función recurrente?

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{3}\right) + g(n) & \text{en otro caso.} \end{cases}$$

Detalles sobre las alternativas de respuesta:

- a) No converge, dependiendo a sus valores iniciales.
- b) Si  $g(n) \in \Theta(1)$  la función de recurrencia representa la complejidad temporal del algoritmo de búsqueda secuencial.
- c) Si  $g(n) \in \Theta(n^2)$  la función de recurrencia representa la complejidad temporal del algoritmo de ordenación burbuja.
- d) Si  $g(n) = \Theta(n)$  la función de recurrencia representa la complejidad temporal del algoritmo de ordenación merge.

Solución: D

De la 85 a la 117 no son seguras :)

Pertenece  $3n^2$  a  $O(n^3)$

Solución: D

Pregunta 11  
Sin responder aún  
Puntúa como 1,00  
1º Muestra pregunta

Pertenece  $3n^2$  a  $O(n^3)$ ?

Seleccione una:

- a. No.
- b. Sí, pero sólo si se toma  $c > 3$ .
- c. No contesto (equivalente a no marcar nada).
- d. Ninguna de las otras dos opciones son verdaderas.

[Quitar mi elección](#)

De la 85 a la 117 no son seguras :)

Tenemos un vector ordenado de tamaño  $n_o$  y un vector desordenado de tamaño  $n_d$

Tenemos un vector ordenado de tamaño  $n_o$  y un vector desordenado de tamaño  $n_d$  y queremos obtener un vector ordenado con todos sus elementos. ¿Qué será más rápido?

Seleccione una:

- a. Ordenar el desordenado y luego mezclar las listas.
- b. Depende de si  $n_o > n_d$  o no.
- c. Insertar los elementos del vector desordenado (uno a uno) en el vector ordenado.
- d. No contesto (equivalente a no marcar nada).

Solución: A

De la 85 a la 117 no son seguras :)

Se pretende obtener la complejidad temporal en el caso más desfavorable de la siguiente función.

Solución: A o D

89

Se pretende obtener la complejidad temporal en el caso más desfavorable de la siguiente función.

```
int suma (int n) {
 int i, sum = 0; n = 0;
 if (n > 0) {
 int j = n;
 while (sum < 100 and j != 0) {
 j = j / 2;
 sum += j;
 }
 if (j == 0) i = n;
 sum += i;
 }
 return sum;
}
```

¿Cuál de las siguientes formulaciones expresa mejor el tiempo?

Selección entre:

a  $r_0(n) = \sum_{i=1}^{\log n} \sum_{j=0}^{2^i-1} \left(\frac{1}{2}\right)^j \in O(n \log n)$

b  $s_0(n) = \sum_{i=0}^{n-1} \left(\frac{1}{2} \sum_{j=0}^i 1\right) \in O(n \log n)$

c No cumple (equivale a no marcar nulla).

d  $t_0(n) = \sum_{i=0}^{\log n} (n - n/2^i) \in O(n \log n)$

De la 85 a la 117 no son seguras :)

La siguiente relación de recurrencia expresa la complejidad de un algoritmo

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Dí cuál de las siguientes afirmaciones es falsa:

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Si  $g(n) \in \Theta(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.
- c. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort.
- d. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación Mergesort.

Solución: C

De la 85 a la 117 no son seguras :)

¿Cuál de las siguientes relaciones de recurrencia expresa el número de llamadas recursivas que hace el algoritmo Mergesort?

¿Cuál de las siguientes relaciones de recurrencia expresa el número de llamadas recursivas que hace el algoritmo Mergesort?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $T(n) = n + T(n/2)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$
- c.  $T(n) = 1 + 2T(n/2)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$
- d.  $T(n) = n + 2T(n/2)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$

Solución: D

De la 85 a la 117 no son seguras :)

Si  $f \in O(g_1)$  y  $f \in O(g_2)$  entonces

Solución: A

Si  $f \in \Theta(g_1)$  y  $f \in \Theta(g_2)$  entonces

Seleccione una:

- a.  $f \in \Theta(g_1 + g_2)$
- b.  $f \notin \Theta(\max(g_1, g_2))$
- c. No contesto (equivalente a no marcar nada).
- d.  $f \in \Theta(g_1 \cdot g_2)$

De la 85 a la 117 no son seguras :)

¿Qué se entiende por el tamaño de problema?

¿Qué se entiende por tamaño del problema?

Seleccione una:

- a. El número de parámetros que componen el problema.
- b. El valor máximo que puede tomar una instancia cualquiera de ese problema.
- c. La cantidad de espacio en memoria que se necesita para codificar una instancia de ese problema.
- d. No contesto (equivalente a no marcar nada).

Solución: C

De la 85 a la 117 no son seguras :)

¿En qué caso la complejidad temporal del algoritmo de ordenación Quicksort es igual a la complejidad temporal del algoritmo Mergesort?

[En qué caso la complejidad temporal del algoritmo de ordenación Quicksort es igual a la complejidad temporal del algoritmo Mergesort?]

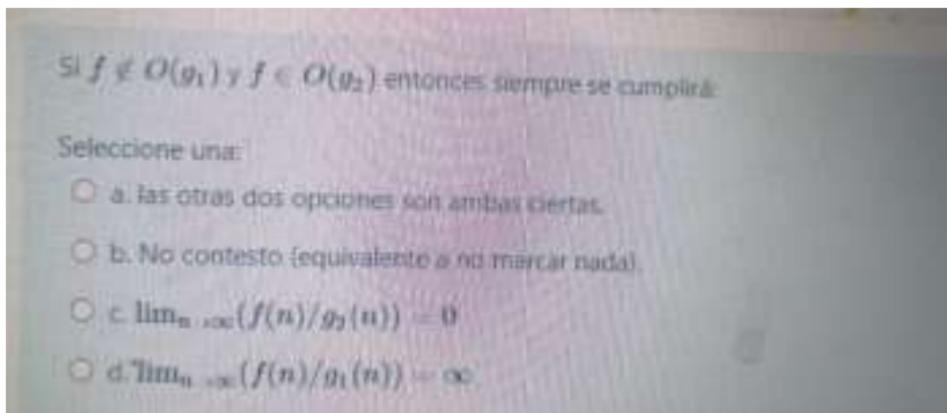
Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. En el caso mejor de ambos.
- c. En el caso peor de ambos.
- d. Tanto en el caso peor como en el caso mejor de ambos.

Solución: B

De la 85 a la 117 no son seguras :)

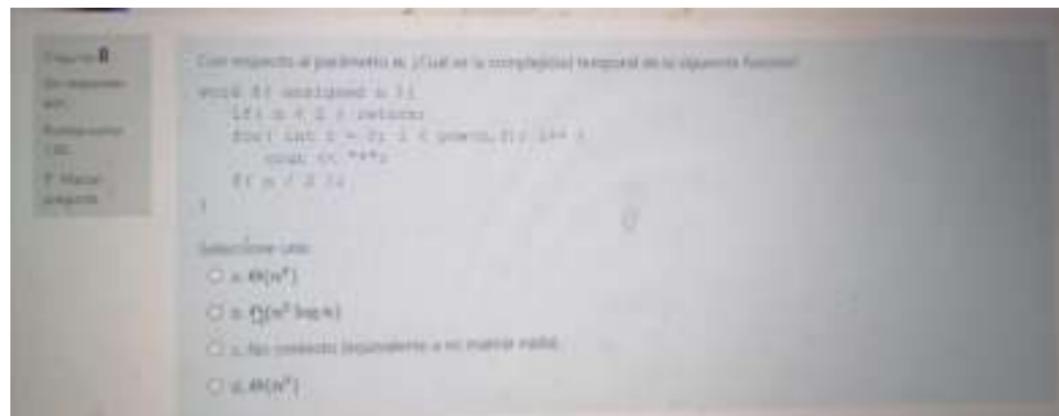
Si  $f \notin O(g_1)$  y  $f \in O(g_2)$  entonces se cumplirá:



Solución: C

De la 85 a la 117 no son seguras :)

Con respecto al parámetro n. ¿Cuál es la complejidad temporal de la siguiente función?



Solución: D

De la 85 a la 117 no son seguras :)

De las siguientes expresiones, o bien dos son verdaderas y una falsa

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.

Selecciona una:

- a.  $\log(n^2) \notin \Theta(\log_d(n))$  ✓
- b. No contesto (equivalente a no marcar nada).
- c.  $\Theta(\log^2(n)) = \Theta(\log^3(n))$  F
- d.  $\Theta(\log_2(n)) = \Theta(\log_3(n))$  ✓

Solución: C

De la 85 a la 117 no son seguras :)

Con respecto a la complejidad temporal de la siguiente función

Con respecto a la complejidad temporal de la siguiente función, ¿cuál de las siguientes afirmaciones es cierta?

```
unsigned long pot2_3(unsigned n) {
 if (n==0)
 return 1;
 if (n&2==0)
 return pot2_3(n/2) * pot2_3(n/2);
 else
 return 2 * pot2_3(n/2) + pot2_3(n/2);
}
```

Seleccione una:

- a. Las otras dos afirmaciones son siempre falsas.
- b. No contiene (equivale a no marcar nada).
- c. La complejidad temporal en el mejor de los casos es constante.
- d. El costo temporal exacto de la función es  $O(n)$ .

Solución: D

De la 85 a la 117 no son seguras :)

Con respecto al parámetro n. ¿Cuál es la complejidad temporal de la siguiente función?

Preguntas 5  
Sí responder  
80%  
Puntuación  
100  
Felicidades  
¡Muy bien!  
Entregar

Con respecto al parámetro n. ¿Cuál es la complejidad temporal de la siguiente función?

```
void f(unsigned n) {
 if (n < 2) return;
 for(int i = 0; i < pow(n, 2); i++)
 cout << i;
 f(n - 2);
}
```

Seleccione una:

- a.  $\Theta(n^2)$
- b.  $\Theta(n^3)$
- c. No contesto (equivale a no marcar nada).
- d.  $\Theta(n^2 \log n)$

Solucion: A

De la 85 a la 117 no son seguras :)

¿Qué tienen en común los algoritmos de ordenación Quicksort y Mergesort?

Pregunta 6

Ser responder  
aún

Puntaje como  
1.00

1º Marcar  
pregunta

¿Qué tienen en común los algoritmos de ordenación Quicksort y Mergesort.

Seleccione una:

- a. La complejidad temporal de la división en subproblemas.
- b. No contesto (equivalente a no marcar nada).
- c. La complejidad temporal de la combinación de las soluciones parciales.
- d. El número de llamadas recursivas que hacen en el mejor de los casos.

[Quitar mi elección](#)

Solución: D

100

De la 85 a la 117 no son seguras :)

Sea la siguiente relación de recurrencia:

Pregunta 8

Sin responder  
aún

Puntuación:  
1,00

■ Marcar  
pregunta

Sea la siguiente relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in O(n^2)$ , ¿en cuál de estos tres casos nos podemos encontrar?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $g(n) = 1$
- c.  $g(n) = n$
- d.  $g(n) = n^2$

Solución: d

De la 85 a la 117 no son seguras :)

¿Qué algoritmo es asintóticamente más rápido, el Quicksort o el Mergesort?

¿Qué algoritmo es asintóticamente más rápido, el Quicksort o el Mergesort?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. como su nombre indica, el Quicksort.
- c. el Mergesort es siempre más rápido o igual (salvo una constante) que el Quicksort.
- d. Los dos son igual de rápidos ya que el coste temporal asintótico de ambos es  $O(n \log(n))$ .

Solución: D

De la 85 a la 117 no son seguras :)

Tenemos un vector desordenado y queremos obtener tres elementos más pequeños.

Tenemos un vector desordenado y queremos obtener los tres elementos más pequeños. (Cuál sería la complejidad temporal más ajustada para hacerlo? (sin pérdida de generalidad puedes suponer que en el vector todos los elementos son distintos))

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Lineal con la longitud del vector
- c. El logaritmo de la longitud del vector
- d. Cuadrática con la longitud del vector

Solución: B

De la 85 a la 117 no son seguras :)

Si entonces

Solución: B

---

Si  $f \in \Omega(g_1) \wedge f \in \Omega(g_2)$  entonces

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $f \in \Omega(g_1 + g_2)$
- c.  $f \in \Omega(g_1 \cdot g_2)$
- d.  $f \notin \Omega(\min(g_1, g_2))$

[Quitar mi elección](#)

De la 85 a la 117 no son seguras :)

La siguiente relación de recurrencia expresa la complejidad de un algoritmo

Pregunta 12

Si responde  
sí

Puntuación  
1.00

T<sub>1</sub>: Marcar  
pregunta

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica.

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Dí cuál de las siguientes afirmaciones es falsa.

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Si  $g(n) \in O(n)$  la relación de recurrencia representa la complejidad temporal en el mejor caso del algoritmo de búsqueda del  $k$ -ésimo elemento más pequeño de un vector (estudiado en clase).
- c. Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenamiento por inserción lombrilla.
- d. Si  $g(n) \in O(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

Solución: C

De la 85 a la 117 no son seguras :)

Si puede pasar que

Solución: D

The screenshot shows a digital exam interface. On the left, a sidebar displays 'Pregunta 1' with the text 'Sin responder aún' and 'Puntuación: 0.00'. Below this are two buttons: 'P. Marcar pregunta' and 'P. Salir'. The main area contains a question and four options labeled a, b, c, and d. The question is: 'Si  $f(n) \in O(n^2)$ , ¿puede pasar que  $f(n) \in O(n^3)$ ?' Below the question is the text 'Seleccione una':

- a. No contesto (equivalente a no marcar nada).
- b. Sólo para valores bajos de n.
- c. No, porque  $n^3 \notin O(n^2)$
- d. Es perfectamente posible, ya que  $O(n^2) \subset O(n^3)$

De la 85 a la 117 no son seguras :)

¿Qué tienen en común el algoritmo que obtiene el  $k$ -ésimo elemento más pequeño?

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?

Seleccione una:

- a. La división del problema en subproblemas.
- b. La combinación de las soluciones a los subproblemas.
- c. No contesto (equivalente a no marcar nada).
- d. El número de llamadas recursivas que se hacen.

Solución: A

De la 85 a la 117 no son seguras :)

Si entonces siempre se cumplirá

Si  $f \notin O(g_1)$  y  $f \in O(g_2)$  entonces siempre se cumplirá:

Seleccione una:

- a.  $f \in \Omega(\min(g_1, g_2))$
- b.  $f \in \Omega(g_1 + g_2)$
- c.  $f \notin O(\max(g_1, g_2))$
- d. No contesto (equivalente a no marcar nada).

Solución: C

De la 85 a la 117 no son seguras :)

Sea la siguiente relación de recurrencia:

Solución: C

Sea la siguiente relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 8T\left(\frac{n}{8}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in \Theta(n^2)$ , ¿en cuál de estos tres casos nos podemos encontrar?

Seleccione una:

- a. No contesto (equivalente a no marcar nada)
- b.  $g(n) = n$
- c.  $g(n) = n^2$
- d.  $g(n) = n^3$

De la 85 a la 117 no son seguras :)

Con respecto a la complejidad temporal de la siguiente función,

Solución: A

Con respecto a la complejidad temporal de la siguiente función, ¿cuál de las siguientes afirmaciones es cierta?

```
unsigned long pot2_1(unsigned n) {
 if (n==0)
 return 1;
 if (n&1==0)
 return pot2_1(n/2)*pot2_1(n/2);
 else
 return 2 * pot2_1(n/2) + pot2_1(n/2);
```

Seleccione una:

- a. El costo temporal exacto de la función es  $O(n)$ .
- b. No es tanto equivalente a no tener cache.
- c. Las otras tres afirmaciones son artificiales.
- d. La complejidad temporal en el mejor de los casos es constante.

De la 85 a la 117 no son seguras :)

Se quieren ordenar d números distintos comprendidos entre 1 y n...

Solución: B

Se quieren ordenar d números distintos comprendidos entre 1 y n. Para ello se usa un vector de n elementos que se reordenan siempre a fondo.

A continuación se responden los d números comprendiendo los valores del elemento del vector de resultados correspondientes a las posiciones dadas.

Para obtener el resultado de la ordenación, suministrando los valores de los elementos del vector de resultados que son true.

¿El vector argumento más rápido (aproximadamente) que el Hérgesheimer?

Salvo caso:

- a. No considera implementación de la función std::sort
- b. Siendo w el log(w) > 3 es menor la memoria que depende de la cantidad de datos
- c. No, ya que este algoritmo ha de recorrer todos los datos al ordenar los resultados
- d. si, ya que el Hérgesheimer es O(n log n) y esto es O(n)

De la 85 a la 117 no son seguras :)

Di cuál de estos resultados de coste temporal asintótico es falsa:

Di cuál de estos resultados de coste temporal asintótico es falsa:

Seleccione una:

- a. La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en  $\Omega(n^2)$ .
- b. La ordenación de un vector usando el algoritmo Quicksort requiere en el peor caso un tiempo en  $\Omega(n^2)$ .
- c. No contesto (equivalente a no marcar nada).
- d. La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en  $O(\log n)$ .

Solución: A

De la 85 a la 117 no son seguras :)

Con respecto a la complejidad espacial de los algoritmos de ordenación Quicksort...

Con respecto a la complejidad espacial de los algoritmos de ordenación Quicksort, Heapsort y Mergesort:

Seleccione una:

- a. Mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante.
- b. La complejidad espacial de todos ellos es lineal con el tamaño del vector a ordenar.
- c. Mergesort y Heapsort tienen complejidad espacial lineal con el tamaño del vector a ordenar, la de Quicksort es constante.
- d. No contesto (equivalente a no marcar nada).

Solución: A

De la 85 a la 117 no son seguras :)

Si podemos decir siempre que

Solución: D

Si  $f(n) \in \Theta(n^2)$ , ¿podemos decir siempre que  $f(n) \in O(n^3)$ ?

Seleccione una:

a. Si, ya que  $n^2 \in \Omega(n^3)$

b. No, ya que  $n^2 \notin O(n^3)$

c. No contesto (equivalente a no marcar nada).

d. Si, ya que  $n^2 \in O(n^3)$

[Quitar mi elección](#)

De la 85 a la 117 no son seguras :)

De las siguientes expresiones, o bien dos son verdaderas y una falsa

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.

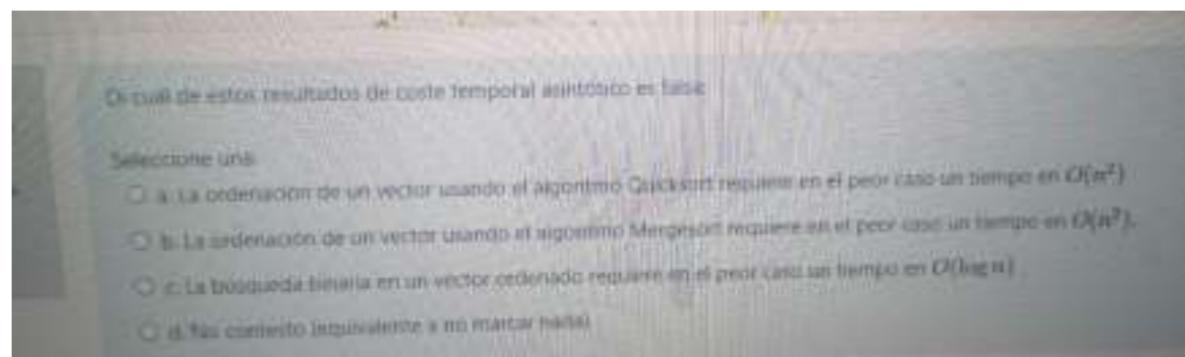
Seleccione una:

- a.  $O(4^{\log(n)}) \subset O(n) \subset O(2^n)$
- b.  $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
- c. No contesto (equivalente a no marcar nada).
- d.  $O(2^{\log(n)}) \subset O(n^2) \subset O(n!)$

Solución: D

De la 85 a la 117 no son seguras :)

Dí cuál de estos resultados de coste temporal asintótico es falsa:



Solución: B

Comenzado el martes, 23 de marzo de 2021, 15:12

Estado Finalizado

Finalizado en martes, 23 de marzo de 2021, 15:28

Tiempo 16 minutos 30 segundos  
empleado

Puntos 6,50/12,00

Calificación 5,42 de 10,00 (54%)

Pregunta 1

Incorrecta

Puntúa -0,50  
sobre 1,00

Con respecto al parámetro  $n$ , ¿Cuál es la complejidad temporal de la siguiente función?

```
void f(unsigned n){
 if(n < 1) return;
 for(int i = 0; i < n; i++)
 for(int j = 0; j < n; j++)
 for(int k = 0; k < n; k++)
 cout << "*";
 for(int i = 0; i < 8; i++)
 f(n / 2);
}
```

Seleccione una:

- a.  $\Theta(n^3)$
- b.  $\Theta(n^2 \log n)$
- c. No contesto (equivalente a no marcar nada).
- d.  $\Theta(n^3 \log n)$



La respuesta correcta es:  $\Theta(n^3 \log n)$

Pregunta 2

Correcta

Puntúa 1,00  
sobre 1,00

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es cierta:

Seleccione una:

- a. Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación mediante inserción binaria.
- b. No contesto (equivalente a no marcar nada).
- c. Si  $g(n) \in O(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.
- d. Si  $g(n) \in O(n)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación Mergesort.



La respuesta correcta es: Si  $g(n) \in O(n)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación Mergesort.

Pregunta 3

Incorrecta

Puntúa 0,00  
sobre 1,00

Cuál de las siguientes formulaciones expresa mejor 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;
}
```

Seleccione una:

- a. Ninguna de las otras dos opciones es correcta.
- b.  $f(n) = \sum_{i=0}^{n/2} \sum_{j=0}^{2*i} 1$ .
- c.  $f(n) = \sum_{i=1}^{\log n} 4n \left(\frac{1}{2}\right)^i$ .
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es:  $f(n) = \sum_{i=1}^{\log n} 4n \left(\frac{1}{2}\right)^i$ .

Pregunta 4

Correcta

Puntúa 1,00  
sobre 1,00

¿Qué se entiende por tamaño del problema?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- 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.
- d. El valor máximo que puede tomar una instancia cualquiera de ese problema.



La respuesta correcta es: La cantidad de espacio en memoria que se necesita para codificar una instancia de ese problema.

Pregunta 5

Correcta

Puntúa 1,00  
sobre 1,00

Sea la siguiente relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in O(n^2)$ , ¿en cuál de estos tres casos nos podemos encontrar?

Seleccione una:

- a.  $g(n) = 1$
- b.  $g(n) = n^2$
- c. No contesto (equivalente a no marcar nada).
- d.  $g(n) = n$

La respuesta correcta es:  $g(n) = n^2$

Pregunta 6

Correcta

Puntúa 1,00  
sobre 1,00

Si  $f(n) \in O(n^3)$ , ¿puede pasar que  $f(n) \in O(n^2)$ ?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Sólo para valores bajos de  $n$
- c. Es perfectamente posible, ya que  $O(n^2) \subset O(n^3)$
- d. No, porque  $n^3 \notin O(n^2)$

La respuesta correcta es: Es perfectamente posible, ya que  $O(n^2) \subset O(n^3)$

Pregunta 7

Correcta

Puntúa 1,00  
sobre 1,00

Tenemos un vector desordenado y queremos obtener los tres elementos más pequeños. ¿Cuál sería la complejidad temporal más ajustada para hacerlo?  
(sin pérdida de generalidad puedes suponer que en el vector todos los elementos son distintos)

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Cuadrática con la longitud del vector
- c. Lineal con la longitud del vector
- d. El logaritmo de la longitud del vector

La respuesta correcta es: Lineal con la longitud del vector

Pregunta 8

Incorrecta

Puntúa -0,50  
sobre 1,00

Cuál es la complejidad espacial del algoritmo Quicksort?

Seleccione una:

- a.  $O(n \log n)$ .
- b.  $O(1)$ .
- c. No contesto (equivalente a no marcar nada).
- d.  $O(n)$ .

La respuesta correcta es:  $O(1)$ .

Pregunta 9

Correcta

Puntúa 1,00  
sobre 1,00

Si  $f \in \Theta(g_1)$  y  $f \in \Theta(g_2)$  entonces

Seleccione una:

- a.  $f \in \Theta(g_1 + g_2)$
- b.  $f \in \Theta(g_1 \cdot g_2)$
- c.  $f \notin \Theta(\max(g_1, g_2))$
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es:  $f \in \Theta(g_1 + g_2)$

Pregunta 10

Correcta

Puntúa 1,00  
sobre 1,00

Los algoritmos de ordenación Quicksort y Mergesort...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... tienen el mismo coste temporal asintótico tanto en el caso peor como en el caso mejor.
- c. ... tienen el mismo coste temporal asintótico en el caso mejor.
- d. ... tienen el mismo coste temporal asintótico en el caso peor.



La respuesta correcta es: ... tienen el mismo coste temporal asintótico en el caso mejor.

Pregunta 11

Correcta

Puntúa 1,00  
sobre 1,00

Las siguientes funciones calculan el valor de la potencia n-ésima de dos. ¿Cuál es más eficiente en cuanto a coste temporal?

```
unsigned long pot2_1(unsigned n){
 if (n==0) return 1;
 if (n%2==0) return pot2_1(n/2) * pot2_1(n/2);
 else return 2 * pot2_1(n/2) * pot2_1(n/2);
}
```

```
unsigned long pot2_2(unsigned n){
 if (n==0) return 1;
 return 2 * pot2_2(n-1);
}
```

Seleccione una:

- a. La primera, pot2\_1(n), es más eficiente que la otra.
- b. No contesto (equivalente a no marcar nada).
- c. La segunda, pot2\_2(n), es más eficiente que la otra.
- d. Las dos funciones son equivalentes en cuanto a coste temporal.



La respuesta correcta es: Las dos funciones son equivalentes en cuanto a coste temporal.

Pregunta 12

Incorrecta

Puntúa -0,50  
sobre 1,00

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.

Seleccione una:

- a.  $\Theta(\log_2(n)) = \Theta(\log_3(n))$
- b.  $\Theta(\log^2(n)) = \Theta(\log^3(n))$
- c.  $\Theta(\log(n^2)) = \Theta(\log(n^3))$
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es:  $\Theta(\log^2(n)) = \Theta(\log^3(n))$

[← Entrenamiento\\_primer\\_parcial](#)

Ir a...



**Comenzado el** martes, 23 de marzo de 2021, 15:12

**Estado** Finalizado

**Finalizado en** martes, 23 de marzo de 2021, 15:31

**Tiempo empleado** 19 minutos 33 segundos

**Puntos** 7,00/12,00

**Calificación** 5,83 de 10,00 (58%)

Pregunta 1

Incorrecta

Puntúa -0,50 sobre 1,00

Con respecto al parámetro  $n$ , ¿Cuál es la complejidad temporal de la siguiente función?

```
void f(unsigned n){
 if(n < 1) return;
 for(int i = 0; i < n; i++)
 for(int j = 0; j < n; j++)
 for(int k = 0; k < n; k++)
 cout << "*";
 for(int i = 0; i < 8; i++)
 f(n / 2);
}
```

Seleccione una:

- a.  $\Theta(n^3)$  ✗
- b.  $\Theta(n^3 \log n)$
- c.  $\Theta(n^2 \log n)$
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es:  $\Theta(n^3 \log n)$

**Pregunta 2**

Correcta

Puntúa 1,00 sobre 1,00

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es falsa:

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Si  $g(n) \in O(n)$  la relación de recurrencia representa la complejidad temporal en el mejor caso del algoritmo de búsqueda del  $k$ -ésimo elemento más pequeño de un vector (estudiado en clase).
- c. Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria. ✓
- d. Si  $g(n) \in O(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

La respuesta correcta es: Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.

**Pregunta 3**

Correcta

Puntúa 1,00 sobre 1,00

Di cuál de estos resultados de coste temporal asintótico es falso:

Seleccione una:

- a. La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en  $O(\log n)$
- b. La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en  $O(n^2)$ . ✓
- c. La ordenación de un vector usando el algoritmo Quicksort requiere en el peor caso un tiempo en  $O(n^2)$
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en  $O(n^2)$ .

**Pregunta 4**

Correcta

Puntúa 1,00 sobre 1,00

Si  $f(n) \in \Theta(n^2)$ , ¿podemos decir siempre que  $f(n) \in O(n^3)$ ?

Seleccione una:

- a. No, ya que  $n^2 \notin O(n^3)$
- b. Sí, ya que  $n^2 \in O(n^3)$  ✓
- c. Sí, ya que  $n^2 \in \Omega(n^3)$
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: Sí, ya que  $n^2 \in O(n^3)$

## Pregunta 5

Incorrecta

Puntúa -0,50 sobre 1,00

Si  $f \notin O(g_1)$  y  $f \in O(g_2)$  entonces NO siempre se cumplirá:

Seleccione una:

- a.  $f \in \Omega(\min(g_1, g_2))$
- b.  $f \in \Omega(g_1 + g_2)$
- c. No contesto (equivalente a no marcar nada).
- d.  $f \in O(\max(g_1, g_2))$



La respuesta correcta es:  $f \in \Omega(g_1 + g_2)$

## Pregunta 6

Incorrecta

Puntúa 0,00 sobre 1,00

Se pretende obtener la complejidad temporal en el caso más desfavorable de la siguiente función.

```
int exa (vector < int > & v) {
 int i, sum = 0, n = v.size();
 if (n > 0) {
 int j = n;
 while (sum < 100 and j != 0) {
 j = j / 2;
 sum = 0;
 for (i = j; i < n; i++)
 sum += v[i];
 }
 return j;
 }
 else return -1;
}
```

¿Cuál de las siguientes formulaciones expresa mejor dicho coste?

Seleccione una:

- a.  $c_s(n) = \sum_{k=1}^{\log n+1} (n - n/2^k) \in O(n \log n)$
- b.  $c_s(n) = \sum_{j=0}^{n/2} \left( \frac{1}{2} \sum_{i=j}^n 1 \right) \in O(n \log n)$
- c. No contesto (equivalente a no marcar nada).
- d.  $c_s(n) = \sum_{j=1}^{\log n} \sum_{i=1}^j \left( \frac{1}{2} \right)^i \in O(n \log n)$



La respuesta correcta es:  $c_s(n) = \sum_{k=1}^{\log n+1} (n - n/2^k) \in O(n \log n)$

## Pregunta 7

Correcta

Puntúa 1,00 sobre 1,00

Con respecto a la complejidad temporal de la siguiente función, ¿cuál de las siguientes afirmaciones es cierta?

```
unsigned long pot2_1(unsigned n) {
 if (n==0)
 return 1;
 if (n%2==0)
 return pot2_1(n/2) * pot2_1(n/2);
 else
 return 2 * pot2_1(n/2) * pot2_1(n/2);
}
```

Seleccione una:

- a. Las otras dos afirmaciones son ambas falsas.
- b. El coste temporal exacto de la función es  $O(n)$ . ✓
- c. No contesto (equivalente a no marcar nada).
- d. La complejidad temporal en el mejor de los casos es constante.

La respuesta correcta es: El coste temporal exacto de la función es  $O(n)$ .

## Pregunta 8

Correcta

Puntúa 1,00 sobre 1,00

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.

Seleccione una:

- a.  $(4^{\log_2(n)}) \subset O(n) \subset O(2^n)$
- b.  $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$  ✓
- c. No contesto (equivalente a no marcar nada).
- d.  $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$

La respuesta correcta es:  $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$

## Pregunta 9

Correcta

Puntúa 1,00 sobre 1,00

Los algoritmos de ordenación Quicksort y Mergesort...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... tienen el mismo coste temporal asintótico en el caso peor.
- c. ... tienen el mismo coste temporal asintótico tanto en el caso peor como en el caso mejor.
- d. ... tienen el mismo coste temporal asintótico en el caso mejor.



La respuesta correcta es: ... tienen el mismo coste temporal asintótico en el caso mejor.

## Pregunta 10

Incorrecta

Puntúa 0,00 sobre 1,00

Tenemos una lista ordenada de tamaño  $n_o$  y una lista desordenada de tamaño  $n_d$ , queremos obtener una lista ordenada con todos los elementos. ¿Cuál sería la complejidad de insertar, uno a uno, todos los elementos de la lista desordenada en la ordenada.

Seleccione una:

- a.  $O(n_d \log n_o)$
- b. No contesto (equivalente a no marcar nada).
- c.  $O(n_o \cdot n_d + n_d^2)$
- d.  $O(n_d \cdot n_o)$



La respuesta correcta es:  $O(n_o \cdot n_d + n_d^2)$

## Pregunta 11

Correcta

Puntúa 1,00 sobre 1,00

¿Cuál de las siguientes relaciones de recurrencia expresa mejor la complejidad espacial es la del algoritmo Mergesort?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $T(n) = n + T(n - 1)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$
- c.  $T(n) = n + 2T(n/2)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$
- d.  $T(n) = n + T(n/2)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$



La respuesta correcta es:  $T(n) = n + 2T(n/2)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$

## Pregunta 12

Correcta

Puntúa 1,00 sobre 1,00

La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

Seleccione una:

- a. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.
- b. ... ninguna de las anteriores es cierta. ✓
- c. ... es la complejidad temporal de las instancias que están en el caso base.
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: ... ninguna de las anteriores es cierta.

[◀ Entrenamiento\\_primer\\_parcial](#)

Ir a...

**Comenzado el** martes, 23 de marzo de 2021, 09:11

**Estado** Finalizado

**Finalizado en** martes, 23 de marzo de 2021, 09:31

**Tiempo** 19 minutos 48 segundos

**empleador**

**Puntos** 4,50/12,00

**Calificación** 3,75 de 10,00 (38%)

Pregunta 1

Correcta

Puntúa 1,00 sobre 1,00

La relación de recurrencia  $T(n) = 1 + T(n - 1)$  si  $n > 1$ ;  $T(1) = 1$  ...

Seleccione una:

- a. Expresa el número de llamadas recursivas que hace el algoritmo Quicksort en el peor de los casos. ✓
- b. No contesto (equivalente a no marcar nada).
- c. Expresa la complejidad temporal asintótica en el peor de los casos del algoritmo de búsqueda binaria.
- d. Ninguna de las otras dos opciones es cierta.

La respuesta correcta es: Expresa el número de llamadas recursivas que hace el algoritmo Quicksort en el peor de los casos.

Pregunta 2

Incorrecta

Puntúa -0,50 sobre 1,00

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es falsa:

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.
- c. Si  $g(n) \in O(n)$  la relación de recurrencia representa la complejidad temporal en el mejor caso del algoritmo de búsqueda del k-ésimo elemento más pequeño de un vector (estudiado en clase). ✗
- d. Si  $g(n) \in O(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

La respuesta correcta es: Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.

Pregunta 3

Incorrecta

Puntúa -0,50 sobre 1,00

Las siguientes funciones calculan el valor de la potencia n-ésima de dos. ¿Cuál es más eficiente en cuanto a coste temporal?

```
unsigned long pot2_1(unsigned n) {
 if (n==0) return 1;
 if (n%2==0) return pot2_1(n/2) * pot2_1(n/2);
 else return 2 * pot2_1(n/2) * pot2_1(n/2);
}

unsigned long pot2_2(unsigned n) {
 if (n==0) return 1;
 return 2 * pot2_2(n-1);
}
```

Seleccione una:

- a. La primera, pot2\_1(n), es más eficiente que la otra. ✗
- b. La segunda, pot2\_2(n), es más eficiente que la otra.
- c. No contesto (equivalente a no marcar nada).
- d. Las dos funciones son equivalentes en cuanto a coste temporal.

La respuesta correcta es: Las dos funciones son equivalentes en cuanto a coste temporal.

Pregunta 4

Correcta

Puntúa 1,00 sobre 1,00

Si  $f_1(n) \in O(g_1(n))$  y  $f_2(n) \in O(g_2(n))$  entonces...

Seleccione una:

- a. Las dos anteriores son ciertas. ✓
- b. No contesto (equivalente a no marcar nada).
- c.  $f_1(n) + f_2(n) \in O(\max(g_1(n), g_2(n)))$
- d.  $f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$

La respuesta correcta es: Las dos anteriores son ciertas.

Pregunta 5

Correcta

Puntúa 1,00 sobre 1,00

La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... ninguna de las anteriores es cierta. ✓
- c. ... es la complejidad temporal de las instancias que están en el caso base.
- d. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.

La respuesta correcta es: ... ninguna de las anteriores es cierta.

Pregunta 6

Correcta

Puntúa 1,00 sobre 1,00

Sea la siguiente relación de recurrencia  $T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$  Si  $T(n) \in O(n)$ , ¿en cuál de estos tres casos nos podemos encontrar?

Seleccione una:

- a.  $g(n) = n$
- b.  $g(n) = n^2$
- c.  $g(n) = 1$  ✓
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es:  $g(n) = 1$

Pregunta 7

Correcta

Puntúa 1,00 sobre 1,00

De las siguientes tres afirmaciones, una es cierta y dos falsas, o bien una es falsa y dos son ciertas. Marca la que en ese sentido es diferente a las otras dos.

Seleccione una:

- a.  $O(2^{\log_2(n)}) \subseteq O(n^2) \subset O(2^n)$
- b. No contesto (equivalente a no marcar nada).
- c.  $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$  ✓
- d.  $O(4^{\log_2(n)}) \subseteq O(n^2) \subset O(2^n)$

La respuesta correcta es:  $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$

Pregunta 8

Correcta

Puntúa 1,00 sobre 1,00

Si  $f(n) \in O(n^3)$ , ¿puede pasar que  $f(n) \in O(n^2)$ ?

Seleccione una:

- a. Sólo para valores bajos de  $n$
- b. No, porque  $n^3 \notin O(n^2)$
- c. Es perfectamente posible, ya que  $O(n^2) \subset O(n^3)$  ✓
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: Es perfectamente posible, ya que  $O(n^2) \subset O(n^3)$

Pregunta **9**

Incorrecta

Puntúa -0,50 sobre 1,00

Con respecto al parámetro  $n$ , ¿Cuál es la complejidad temporal de la siguiente función?

```
void f(unsigned n) {
 if(n < 2) return;
 for(int i = 0; i < pow(n,3); i++)
 cout << "*";
 f(n / 2);
}
```

Seleccione una:

- a.  $\Theta(n^3 \log n)$
- b.  $\Theta(n^4)$
- c.  $\Theta(n^3)$
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es:  $\Theta(n^3)$

Pregunta **10**

Correcta

Puntúa 1,00 sobre 1,00

Los algoritmos de ordenación Quicksort y Mergesort tienen en común ...

Seleccione una:

- a. ... que aplican la estrategia de divide y vencerás.
- b. No contesto (equivalente a no marcar nada).
- c. ... que se ejecutan en tiempo  $O(n)$ .
- d. ... que ordenan el vector sin usar espacio adicional.



La respuesta correcta es: ... que aplican la estrategia de divide y vencerás.

Pregunta 11

Incorrecta

Puntúa -0,50 sobre 1,00

Tenemos un vector ordenado de tamaño  $n_o$  y un vector desordenado de tamaño  $n_d$  y queremos obtener un vector ordenado con todos los elementos. ¿Qué será más rápido?

Seleccione una:

- a. Insertar los elementos del vector desordenado (uno a uno) en el vector ordenado.
- b. No contesto (equivalente a no marcar nada).
- c. Depende de si  $n_o > n_d$  o no. ✗
- d. Ordenar el desordenado y luego mezclar las listas.

La respuesta correcta es: Ordenar el desordenado y luego mezclar las listas.

Pregunta 12

Incorrecta

Puntúa -0,50 sobre 1,00

Se pretende obtener la complejidad temporal en el caso más desfavorable de la siguiente función.

```
int exa (vector < int > & v) {
 int i, sum = 0, n = v.size();
 if (n > 0) {
 int j = n;
 while (sum < 100 and j != 0) {
 j = j / 2;
 sum = 0;
 for (i = j; i < n; i++)
 sum += v[i];
 }
 return j;
 }
 else return -1;
}
```

¿Cuál de las siguientes formulaciones expresa dicho coste?

Seleccione una:

- a.  $c_s(n) = \sum_{k=1}^{\log n+1} (n - n/2^k) \in O(n \log n)$  ✗
- b. No contesto (equivalente a no marcar nada).
- c. Las otras dos opciones son ambas ciertas.
- d.  $c_s(n) = n \sum_{j=1}^{\log n} \sum_{i=1}^j \left(\frac{1}{2}\right)^i \in O(n \log n)$

La respuesta correcta es: Las otras dos opciones son ambas ciertas.

[◀ Entrenamiento\\_primer\\_parcial](#)

Ir a...

[Área personal](#) / Mis cursos / [34018\\_2020-21](#) / [Primer examen parcial de ADA](#) / [Primer parcial](#)

**Comenzado el** martes, 23 de marzo de 2021, 15:12

**Estado** Finalizado

**Finalizado en** martes, 23 de marzo de 2021, 15:29

**Tiempo empleado** 17 minutos 25 segundos

**Puntos** 4,50/12,00

**Calificación** **3,75** de 10,00 (38%)

Pregunta 1

Correcta

Puntúa 1,00 sobre 1,00

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es cierta:

Seleccione una:

- a. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación Mergesort. ✓
- b. Si  $g(n) \in \Theta(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación mediante inserción binaria.
- c. No contesto (equivalente a no marcar nada).
- d. Si  $g(n) \in \Theta(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

La respuesta correcta es: Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación Mergesort.

Pregunta 2

Incorrecta

Puntúa 0,00 sobre 1,00

Con respecto a la complejidad espacial de los algoritmos de ordenación Quicksort, Heapsort y Mergesort:

Seleccione una:

- a. No contesto (equivalente a no marcar nada). ✗
- b. Mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante.
- c. Mergesort y Heapsort tienen complejidad espacial lineal con el tamaño del vector a ordenar, la de Quicksort es constante.
- d. Las complejidad espacial de todos ellos es lineal con el tamaño del vector a ordenar.

La respuesta correcta es: Mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante.

## Pregunta 3

Incorrecta

Puntúa 0,00 sobre 1,00

Si  $f \notin O(g_1)$  y  $f \in O(g_2)$  entonces siempre se cumplirá:

Seleccione una:

- a.  $\lim_{n \rightarrow \infty} (f(n)/g_1(n)) = \infty$
- b. las otras dos opciones son ambas ciertas.
- c. No contesto (equivalente a no marcar nada). X
- d.  $\lim_{n \rightarrow \infty} (f(n)/g_2(n)) = 0$

La respuesta correcta es:  $\lim_{n \rightarrow \infty} (f(n)/g_1(n)) = \infty$

## Pregunta 4

Incorrecta

Puntúa 0,00 sobre 1,00

Cuál de las siguientes formulaciones expresa mejor 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;
}
```

Seleccione una:

- a.  $f(n) = \sum_{i=0}^{n/2} \sum_{j=0}^{2*i} 1.$
- b.  $f(n) = \sum_{i=1}^{\log n} 4n \left(\frac{1}{2}\right)^i.$
- c. Ninguna de las otras dos opciones es correcta.
- d. No contesto (equivalente a no marcar nada). X

La respuesta correcta es:  $f(n) = \sum_{i=1}^{\log n} 4n \left(\frac{1}{2}\right)^i.$

## Pregunta 5

Correcta

Puntúa 1,00 sobre 1,00

Pertenece  $3n^2 + 3$  a  $O(n^3)$ ?

Seleccione una:

- a. No.
  - b. Solo para  $c = 6$  y  $n_0 = 1$
  - c. Sí.
  - d. No contesto (equivalente a no marcar nada).
- ✓

La respuesta correcta es: Sí.

## Pregunta 6

Correcta

Puntúa 1,00 sobre 1,00

Sea la siguiente relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in O(n^2)$ , ¿en cuál de estos tres casos nos podemos encontrar?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
  - b.  $g(n) = n^2$
  - c.  $g(n) = n \log n$
  - d.  $g(n) = n$
- ✓

La respuesta correcta es:  $g(n) = n^2$

## Pregunta 7

Correcta

Puntúa 1,00 sobre 1,00

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.

Seleccione una:

- a.  $\Theta(\log^2(n)) = \Theta(\log^3(n))$
  - b. No contesto (equivalente a no marcar nada).
  - c.  $\Theta(\log_2(n)) = \Theta(\log_3(n))$
  - d.  $\Theta(\log(n^2)) = \Theta(\log(n^3))$
- ✓

La respuesta correcta es:  $\Theta(\log^2(n)) = \Theta(\log^3(n))$

## Pregunta 8

Incorrecta

Puntúa -0,50 sobre 1,00

Con respecto al parámetro  $n$ , ¿Cuál es la complejidad temporal de la siguiente función?

```
void f(unsigned n){
 if(n < 1) return;
 for(int i = 0; i < n; i++)
 for(int j = 0; j < n; j++)
 for(int k = 0; k < n; k++)
 cout << "*";
 for(int i = 0; i < 8; i++)
 f(n / 2);
}
```

Seleccione una:

- a.  $\Theta(n^2 \log n)$
- b. No contesto (equivalente a no marcar nada).
- c.  $\Theta(n^3 \log n)$
- d.  $\Theta(n^3)$



La respuesta correcta es:  $\Theta(n^3 \log n)$

## Pregunta 9

Correcta

Puntúa 1,00 sobre 1,00

Con respecto a la complejidad temporal de la siguiente función, ¿cuál de las siguientes afirmaciones es cierta?

```
unsigned long pot2_1(unsigned n) {
 if (n==0)
 return 1;
 if (n%2==0)
 return pot2_1(n/2) * pot2_1(n/2);
 else
 return 2 * pot2_1(n/2) * pot2_1(n/2);
}
```

Seleccione una:

- a. La complejidad temporal en el mejor de los casos es constante.
- b. Las otras dos afirmaciones son ambas falsas.
- c. El coste temporal exacto de la función es  $O(n)$ .
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es: El coste temporal exacto de la función es  $O(n)$ .

**Pregunta 10**

Incorrecta

Puntúa -0,50 sobre 1,00

La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

Seleccione una:

- a. ... es la complejidad temporal de las instancias que están en el caso base. ✗
- b. No contesto (equivalente a no marcar nada).
- c. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.
- d. ... ninguna de las anteriores es cierta.

La respuesta correcta es: ... ninguna de las anteriores es cierta.

**Pregunta 11**

Correcta

Puntúa 1,00 sobre 1,00

Tenemos un vector desordenado y queremos obtener los tres elementos más pequeños. ¿Cuál sería la complejidad temporal más ajustada para hacerlo?

(sin pérdida de generalidad puedes suponer que en el vector todos los elementos son distintos)

Seleccione una:

- a. El logaritmo de la longitud del vector
- b. Cuadrática con la longitud del vector
- c. Lineal con la longitud del vector ✓
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: Lineal con la longitud del vector

**Pregunta 12**

Incorrecta

Puntúa -0,50 sobre 1,00

¿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?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. La división del problema en subproblemas.
- c. El número de llamadas recursivas que se hacen. ✗
- d. La combinación de las soluciones a los subproblemas.

La respuesta correcta es: La división del problema en subproblemas.

[◀ Entrenamiento\\_primer\\_parcial](#)

[Ir a...](#)

Área personal / Mis cursos / [34018 2020-21](#) / [Primer examen parcial de ADA](#) / [Primer parcial](#)

**Comenzado el** martes, 23 de marzo de 2021, 09:11

**Estado** Finalizado

**Finalizado en** martes, 23 de marzo de 2021, 09:30

**Tiempo empleado** 19 minutos 29 segundos

**Puntos** -0,50/12,00

**Calificación** **-0,42** de 10,00 (-4%)

Pregunta 1

Incorrecta

Puntúa -0,50 sobre 1,00

¿Qué se entiende por tamaño del problema?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. El valor máximo que puede tomar una instancia cualquiera de ese problema. ✗
- c. El número de parámetros que componen el problema.
- d. La cantidad de espacio en memoria que se necesita para codificar una instancia de ese problema.

La respuesta correcta es: La cantidad de espacio en memoria que se necesita para codificar una instancia de ese problema.

Pregunta 2

Incorrecta

Puntúa -0,50 sobre 1,00

De las siguientes tres afirmaciones, una es cierta y dos falsas, o bien una es falsa y dos son ciertas. Marca la que en ese sentido es diferente a las otras dos.

Seleccione una:

- a.  $O(2^{\log_2(n)}) \subseteq O(n^2) \subset O(2^n)$
- b. No contesto (equivalente a no marcar nada).
- c.  $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$
- d.  $O(4^{\log_2(n)}) \subseteq O(n^2) \subset O(2^n)$  ✗

La respuesta correcta es:  $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$

## Pregunta 3

Incorrecta

Puntúa -0,50 sobre 1,00

Di cuál de estos resultados de coste temporal asintótico es falsa:

Seleccione una:

- a. La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en  $O(\log n)$
- b. La ordenación de un vector usando el algoritmo Quicksort requiere en el peor caso un tiempo en  $O(n^2)$  ✗
- c. La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en  $O(n^2)$ .
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en  $O(n^2)$ .

## Pregunta 4

Incorrecta

Puntúa 0,00 sobre 1,00

Se pretende obtener la complejidad temporal en el caso más desfavorable de la siguiente función.

```
int exa (vector < int > & v) {
 int i, sum = 0, n = v.size();
 if (n > 0) {
 int j = n;
 while (sum < 100 and j != 0) {
 j = j / 2;
 sum = 0;
 for (i = j; i < n; i++)
 sum += v[i];
 }
 return j;
 }
 else return -1;
}
```

¿Cuál de las siguientes formulaciones expresa dicho coste?

Seleccione una:

- a.  $c_s(n) = \sum_{k=1}^{\log n + 1} (n - n/2^k) \in O(n \log n)$
- b. No contesto (equivalente a no marcar nada). ✗
- c.  $c_s(n) = n \sum_{j=1}^{\log n} \sum_{i=1}^j \left(\frac{1}{2}\right)^i \in O(n \log n)$
- d. Las otras dos opciones son ambas ciertas.

La respuesta correcta es: Las otras dos opciones son ambas ciertas.

## Pregunta 5

Correcta

Puntúa 1,00 sobre 1,00

Tenemos una lista ordenada de tamaño  $n_o$  y una lista desordenada de tamaño  $n_d$ , queremos obtener una lista ordenada con todos los elementos. ¿Cuál sería la complejidad de insertar, uno a uno, todos los elementos de la lista desordenada en la ordenada.

Seleccione una:

- a.  $O(n_o \cdot n_d + n_d^2)$  ✓
- b.  $O(n_d \log n_o)$
- c.  $O(n_d \cdot n_o)$
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es:  $O(n_o \cdot n_d + n_d^2)$

## Pregunta 6

Incorrecta

Puntúa -0,50 sobre 1,00

Si  $f \in \Theta(g_1)$  y  $f \in \Theta(g_2)$  entonces

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $f \in \Theta(g_1 \cdot g_2)$  ✗
- c.  $f \in \Theta(g_1 + g_2)$
- d.  $f \notin \Theta(\max(g_1, g_2))$

La respuesta correcta es:  $f \in \Theta(g_1 + g_2)$

## Pregunta 7

Incorrecta

Puntúa -0,50 sobre 1,00

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es falsa:

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Si  $g(n) \in O(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.
- c. Si  $g(n) \in O(n)$  la relación de recurrencia representa la complejidad temporal en el mejor caso del algoritmo de búsqueda del k-ésimo elemento más pequeño de un vector (estudiado en clase). ✗
- d. Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.

La respuesta correcta es: Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.

## Pregunta 8

Correcta

Puntúa 1,00 sobre 1,00

Los algoritmos de ordenación Quicksort y Mergesort...

Seleccione una:

- a. ... tienen el mismo coste temporal asintótico en el caso mejor. ✓
- b. ... tienen el mismo coste temporal asintótico en el caso peor.
- c. No contesto (equivalente a no marcar nada).
- d. ... tienen el mismo coste temporal asintótico tanto en el caso peor como en el caso mejor.

La respuesta correcta es: ... tienen el mismo coste temporal asintótico en el caso mejor.

## Pregunta 9

Incorrecta

Puntúa 0,00 sobre 1,00

Con respecto a la complejidad temporal de la siguiente función, ¿cuál de las siguientes afirmaciones es cierta?

```
unsigned long pot2_1(unsigned n) {
 if (n==0)
 return 1;
 if (n%2==0)
 return pot2_1(n/2) * pot2_1(n/2);
 else
 return 2 * pot2_1(n/2) * pot2_1(n/2);
}
```

Seleccione una:

- a. La complejidad temporal en el mejor de los casos es constante.
- b. Las otras dos afirmaciones son ambas falsas.
- c. El coste temporal exacto de la función es  $O(n)$ . ✗
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: El coste temporal exacto de la función es  $O(n)$ .

## Pregunta 10

Correcta

Puntúa 1,00 sobre 1,00

Con respecto al parámetro  $n$ , ¿Cuál es la complejidad temporal de la siguiente función?

```
void f(unsigned n){
 if(n < 1) return;
 for(int i = 0; i < n; i++)
 for(int j = 0; j < n; j++)
 for(int k = 0; k < n; k++)
 cout << "*";
 for(int i = 0; i < 8; i++)
 f(n / 2);
}
```

Seleccione una:

- a.  $\Theta(n^2 \log n)$
- b.  $\Theta(n^3)$
- c. No contesto (equivalente a no marcar nada).
- d.  $\Theta(n^3 \log n)$



La respuesta correcta es:  $\Theta(n^3 \log n)$

## Pregunta 11

Incorrecta

Puntúa -0,50 sobre 1,00

Sea la siguiente relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in O(n^2)$ , ¿en cuál de estos tres casos nos podemos encontrar?

Seleccione una:

- a.  $g(n) = n$
- b.  $g(n) = n \log n$
- c.  $g(n) = n^2$
- d. No contesto (equivalente a no marcar nada).



La respuesta correcta es:  $g(n) = n^2$



## Pregunta 12

Incorrecta

Puntúa -0,50 sobre 1,00

Si  $f(n) \in O(n^3)$ , ¿puede pasar que  $f(n) \in O(n^2)$ ?

Seleccione una:

- a. Es perfectamente posible, ya que  $O(n^2) \subset O(n^3)$
- b. Sólo para valores bajos de  $n$
- c. No, porque  $n^3 \notin O(n^2)$  X
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: Es perfectamente posible, ya que  $O(n^2) \subset O(n^3)$

[◀ Entrenamiento\\_primer\\_parcial](#)

Ir a...



**Comenzado el** martes, 23 de marzo de 2021, 09:11

**Estado** Finalizado

**Finalizado en** martes, 23 de marzo de 2021, 09:31

**Tiempo empleado** 20 minutos

**Puntos** 7,00/12,00

**Calificación** 5,83 de 10,00 (58%)

Pregunta 1

Incorrecta

Puntúa 0,00 sobre 1,00

Se pretende obtener la complejidad temporal en el caso más desfavorable de la siguiente función.

```
int exa (vector < int > & v) {
 int i, sum = 0, n = v.size();
 if (n > 0) {
 int j = n;
 while (sum < 100 and j != 0) {
 j = j / 2;
 sum = 0;
 for (i = j; i < n; i++)
 sum += v[i];
 }
 return j;
 }
 else return -1;
}
```

¿Cuál de las siguientes formulaciones expresa dicho coste?

Seleccione una:

- a. No contesto (equivalente a no marcar nada). ✖
- b.  $c_s(n) = \sum_{k=1}^{\log n + 1} (n - n/2^k) \in O(n \log n)$
- c.  $c_s(n) = n \sum_{j=1}^{\log n} \sum_{i=1}^j \left(\frac{1}{2}\right)^i \in O(n \log n)$
- d. Las otras dos opciones son ambas ciertas.

La respuesta correcta es: Las otras dos opciones son ambas ciertas.



## Pregunta 2

Correcta

Puntúa 1,00 sobre 1,00

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.

Seleccione una:

- a.  $\log(n^3) \notin \Theta(\log_3(n))$
- b. No contesto (equivalente a no marcar nada).
- c.  $\Theta(\log^2(n)) = \Theta(\log^3(n))$
- d.  $\Theta(\log_2(n)) = \Theta(\log_3(n))$



La respuesta correcta es:  $\Theta(\log_2(n)) = \Theta(\log_3(n))$

## Pregunta 3

Incorrecta

Puntúa -0,50 sobre 1,00

Con respecto a la complejidad temporal de la siguiente función, ¿cuál de las siguientes afirmaciones es cierta?

```
unsigned long pot2_1(unsigned n) {
 if (n==0)
 return 1;
 if (n%2==0)
 return pot2_1(n/2) * pot2_1(n/2);
 else
 return 2 * pot2_1(n/2) * pot2_1(n/2);
}
```

Seleccione una:

- a. Las otras dos afirmaciones son ambas falsas. ✗
- b. No contesto (equivalente a no marcar nada).
- c. La complejidad temporal en el mejor de los casos es constante.
- d. El coste temporal exacto de la función es  $O(n)$ .

La respuesta correcta es: El coste temporal exacto de la función es  $O(n)$ .



**Pregunta 4**

Correcta

Puntúa 1,00 sobre 1,00

Los algoritmos de ordenación Quicksort y Mergesort tienen en común ...

Seleccione una:

- a. ... que ordenan el vector sin usar espacio adicional.
- b. No contesto (equivalente a no marcar nada).
- c. ... que aplican la estrategia de divide y vencerás.
- d. ... que se ejecutan en tiempo  $O(n)$ .



La respuesta correcta es: ... que aplican la estrategia de divide y vencerás.

**Pregunta 5**

Correcta

Puntúa 1,00 sobre 1,00

Con respecto al parámetro  $n$ , ¿Cuál es la complejidad temporal de la siguiente función?

```
void f(unsigned n) {
 if(n < 2) return;
 for(int i = 0; i < pow(n,2); i++)
 cout << "*";
 f(n - 2);
}
```

Seleccione una:

- a.  $\Theta(n^2)$
- b.  $\Theta(n^3)$
- c. No contesto (equivalente a no marcar nada).
- d.  $\Theta(n^2 \log n)$



La respuesta correcta es:  $\Theta(n^3)$

**Pregunta 6**

Correcta

Puntúa 1,00 sobre 1,00

¿Qué tienen en común los algoritmos de ordenación Quicksort y Mergesort.

Seleccione una:

- a. La complejidad temporal de la división en subproblemas.
- b. No contesto (equivalente a no marcar nada).
- c. La complejidad temporal de la combinación de las soluciones parciales.
- d. El número de llamadas recursivas que hacen en el mejor de los casos.



La respuesta correcta es: El número de llamadas recursivas que hacen en el mejor de los casos.

## Pregunta 7

Correcta

Puntúa 1,00 sobre 1,00

Si  $f(n) \in O(n^2)$ , ¿podemos decir siempre que  $f(n) \in O(n^3)$ ?

Seleccione una:

- a. Sí ya que  $n^2 \in O(n^3)$
- b. No, ya que  $n^2 \notin O(n^3)$
- c. No contesto (equivalente a no marcar nada).
- d. Sólo para valores bajos de  $n$



La respuesta correcta es: Sí ya que  $n^2 \in O(n^3)$

## Pregunta 8

Correcta

Puntúa 1,00 sobre 1,00

Sea la siguiente relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in O(n^2)$ , ¿en cuál de estos tres casos nos podemos encontrar?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $g(n) = 1$
- c.  $g(n) = n$
- d.  $g(n) = n^2$



La respuesta correcta es:  $g(n) = n^2$

## Pregunta 9

Incorrecta

Puntúa -0,50 sobre 1,00

Si  $f \in \Omega(g_1)$  y  $f \in \Omega(g_2)$  entonces

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $f \in \Omega(g_1 + g_2)$
- c.  $f \in \Omega(g_1 \cdot g_2)$
- d.  $f \notin \Omega(\min(g_1, g_2))$



La respuesta correcta es:  $f \in \Omega(g_1 + g_2)$

**Pregunta 10**

Correcta

Puntúa 1,00 sobre 1,00

Tenemos un vector ordenado de tamaño  $n_o$  y un vector desordenado de tamaño  $n_d$  y queremos obtener un vector ordenado con todos los elementos. ¿Qué será más rápido?

Seleccione una:

- a. Depende de si  $n_o > n_d$  o no.
- b. Ordenar el desordenado y luego mezclar las listas. ✓
- c. No contesto (equivalente a no marcar nada).
- d. Insertar los elementos del vector desordenado (uno a uno) en el vector ordenado.

La respuesta correcta es: Ordenar el desordenado y luego mezclar las listas.

**Pregunta 11**

Correcta

Puntúa 1,00 sobre 1,00

La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

Seleccione una:

- a. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.
- b. ... es la complejidad temporal de las instancias que están en el caso base.
- c. ... ninguna de las anteriores es cierta. ✓
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: ... ninguna de las anteriores es cierta.

**Pregunta 12**

Sin contestar

Puntúa como 1,00

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es falsa:

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Si  $g(n) \in O(n)$  la relación de recurrencia representa la complejidad temporal en el mejor caso del algoritmo de búsqueda del k-ésimo elemento más pequeño de un vector (estudiado en clase).
- c. Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.
- d. Si  $g(n) \in O(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

La respuesta correcta es: Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.

[◀ Entrenamiento\\_primer\\_parcial](#)

Ir a...



**Comenzado el** martes, 23 de marzo de 2021, 09:11

**Estado** Finalizado

**Finalizado en** martes, 23 de marzo de 2021, 09:31

**Tiempo empleado** 20 minutos

**Puntos** 7,00/12,00

**Calificación** 5,83 de 10,00 (58%)

Pregunta 1

Incorrecta

Puntúa 0,00 sobre 1,00

Se pretende obtener la complejidad temporal en el caso más desfavorable de la siguiente función.

```
int exa (vector < int > & v) {
 int i, sum = 0, n = v.size();
 if (n > 0) {
 int j = n;
 while (sum < 100 and j != 0) {
 j = j / 2;
 sum = 0;
 for (i = j; i < n; i++)
 sum += v[i];
 }
 return j;
 }
 else return -1;
}
```

¿Cuál de las siguientes formulaciones expresa dicho coste?

Seleccione una:

- a. No contesto (equivalente a no marcar nada). ✖
- b.  $c_s(n) = \sum_{k=1}^{\log n + 1} (n - n/2^k) \in O(n \log n)$
- c.  $c_s(n) = n \sum_{j=1}^{\log n} \sum_{i=1}^j \left(\frac{1}{2}\right)^i \in O(n \log n)$
- d. Las otras dos opciones son ambas ciertas.

La respuesta correcta es: Las otras dos opciones son ambas ciertas.



## Pregunta 2

Correcta

Puntúa 1,00 sobre 1,00

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.

Seleccione una:

- a.  $\log(n^3) \notin \Theta(\log_3(n))$
- b. No contesto (equivalente a no marcar nada).
- c.  $\Theta(\log^2(n)) = \Theta(\log^3(n))$
- d.  $\Theta(\log_2(n)) = \Theta(\log_3(n))$



La respuesta correcta es:  $\Theta(\log_2(n)) = \Theta(\log_3(n))$

## Pregunta 3

Incorrecta

Puntúa -0,50 sobre 1,00

Con respecto a la complejidad temporal de la siguiente función, ¿cuál de las siguientes afirmaciones es cierta?

```
unsigned long pot2_1(unsigned n) {
 if (n==0)
 return 1;
 if (n%2==0)
 return pot2_1(n/2) * pot2_1(n/2);
 else
 return 2 * pot2_1(n/2) * pot2_1(n/2);
}
```

Seleccione una:

- a. Las otras dos afirmaciones son ambas falsas. ✗
- b. No contesto (equivalente a no marcar nada).
- c. La complejidad temporal en el mejor de los casos es constante.
- d. El coste temporal exacto de la función es  $O(n)$ .

La respuesta correcta es: El coste temporal exacto de la función es  $O(n)$ .



**Pregunta 4**

Correcta

Puntúa 1,00 sobre 1,00

Los algoritmos de ordenación Quicksort y Mergesort tienen en común ...

Seleccione una:

- a. ... que ordenan el vector sin usar espacio adicional.
- b. No contesto (equivalente a no marcar nada).
- c. ... que aplican la estrategia de divide y vencerás.
- d. ... que se ejecutan en tiempo  $O(n)$ .



La respuesta correcta es: ... que aplican la estrategia de divide y vencerás.

**Pregunta 5**

Correcta

Puntúa 1,00 sobre 1,00

Con respecto al parámetro  $n$ , ¿Cuál es la complejidad temporal de la siguiente función?

```
void f(unsigned n) {
 if(n < 2) return;
 for(int i = 0; i < pow(n,2); i++)
 cout << "*";
 f(n - 2);
}
```

Seleccione una:

- a.  $\Theta(n^2)$
- b.  $\Theta(n^3)$
- c. No contesto (equivalente a no marcar nada).
- d.  $\Theta(n^2 \log n)$



La respuesta correcta es:  $\Theta(n^3)$

**Pregunta 6**

Correcta

Puntúa 1,00 sobre 1,00

¿Qué tienen en común los algoritmos de ordenación Quicksort y Mergesort.

Seleccione una:

- a. La complejidad temporal de la división en subproblemas.
- b. No contesto (equivalente a no marcar nada).
- c. La complejidad temporal de la combinación de las soluciones parciales.
- d. El número de llamadas recursivas que hacen en el mejor de los casos.



La respuesta correcta es: El número de llamadas recursivas que hacen en el mejor de los casos.

## Pregunta 7

Correcta

Puntúa 1,00 sobre 1,00

Si  $f(n) \in O(n^2)$ , ¿podemos decir siempre que  $f(n) \in O(n^3)$ ?

Seleccione una:

- a. Sí ya que  $n^2 \in O(n^3)$
- b. No, ya que  $n^2 \notin O(n^3)$
- c. No contesto (equivalente a no marcar nada).
- d. Sólo para valores bajos de  $n$



La respuesta correcta es: Sí ya que  $n^2 \in O(n^3)$

## Pregunta 8

Correcta

Puntúa 1,00 sobre 1,00

Sea la siguiente relación de recurrencia:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in O(n^2)$ , ¿en cuál de estos tres casos nos podemos encontrar?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $g(n) = 1$
- c.  $g(n) = n$
- d.  $g(n) = n^2$



La respuesta correcta es:  $g(n) = n^2$

## Pregunta 9

Incorrecta

Puntúa -0,50 sobre 1,00

Si  $f \in \Omega(g_1)$  y  $f \in \Omega(g_2)$  entonces

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $f \in \Omega(g_1 + g_2)$
- c.  $f \in \Omega(g_1 \cdot g_2)$
- d.  $f \notin \Omega(\min(g_1, g_2))$



La respuesta correcta es:  $f \in \Omega(g_1 + g_2)$

**Pregunta 10**

Correcta

Puntúa 1,00 sobre 1,00

Tenemos un vector ordenado de tamaño  $n_o$  y un vector desordenado de tamaño  $n_d$  y queremos obtener un vector ordenado con todos los elementos. ¿Qué será más rápido?

Seleccione una:

- a. Depende de si  $n_o > n_d$  o no.
- b. Ordenar el desordenado y luego mezclar las listas. ✓
- c. No contesto (equivalente a no marcar nada).
- d. Insertar los elementos del vector desordenado (uno a uno) en el vector ordenado.

La respuesta correcta es: Ordenar el desordenado y luego mezclar las listas.

**Pregunta 11**

Correcta

Puntúa 1,00 sobre 1,00

La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

Seleccione una:

- a. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.
- b. ... es la complejidad temporal de las instancias que están en el caso base.
- c. ... ninguna de las anteriores es cierta. ✓
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: ... ninguna de las anteriores es cierta.

**Pregunta 12**

Sin contestar

Puntúa como 1,00

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es falsa:

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Si  $g(n) \in O(n)$  la relación de recurrencia representa la complejidad temporal en el mejor caso del algoritmo de búsqueda del k-ésimo elemento más pequeño de un vector (estudiado en clase).
- c. Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.
- d. Si  $g(n) \in O(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

La respuesta correcta es: Si  $g(n) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.

[◀ Entrenamiento\\_primer\\_parcial](#)

Ir a...



[Área personal](#) / Mis cursos / [34018 2020-21](#) / [Primer examen parcial de ADA](#) / [Primer parcial](#)

**Comenzado el** martes, 23 de marzo de 2021, 09:10

**Estado** Finalizado

**Finalizado en** martes, 23 de marzo de 2021, 09:30

**Tiempo empleado** 19 minutos 12 segundos

**Puntos** 4,50/12,00

**Calificación** **3,75** de 10,00 (38%)

Pregunta 1

Correcta

Puntúa 1,00 sobre 1,00

Pertenece  $3n^2 + 3$  a  $O(n^3)$ ?

Seleccione una:

- a. No.
- b. Sí. ✓
- c. No contesto (equivalente a no marcar nada).
- d. Solo para  $c = 6$  y  $n_0 = 1$

La respuesta correcta es: Sí.

Pregunta 2

Correcta

Puntúa 1,00 sobre 1,00

La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

Seleccione una:

- a. ... es la complejidad temporal de las instancias que están en el caso base.
- b. ... ninguna de las anteriores es cierta. ✓
- c. No contesto (equivalente a no marcar nada).
- d. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.

La respuesta correcta es: ... ninguna de las anteriores es cierta.

## Pregunta 3

Correcta

Puntúa 1,00 sobre 1,00

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.

Seleccione una:

- a.  $(4^{\log_2(n)}) \subset O(n) \subset O(2^n)$
- b.  $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$
- c.  $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$  ✓
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es:  $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$

## Pregunta 4

Incorrecta

Puntúa -0,50 sobre 1,00

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es falsa:

Seleccione una:

- a. Si  $g(n) \in \Theta(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.
- b. No contesto (equivalente a no marcar nada).
- c. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort.
- d. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación Mergesort. ✗

La respuesta correcta es: Si  $g(n) \in \Theta(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

**Pregunta 5**

Incorrecta

Puntúa -0,50 sobre 1,00

Las siguientes funciones calculan el valor de la potencia n-ésima de dos. ¿Cuál es más eficiente en cuanto a coste temporal?

```
unsigned long pot2_1(unsigned n) {
 if (n==0) return 1;
 if (n%2==0) return pot2_1(n/2) * pot2_1(n/2);
 else return 2 * pot2_1(n/2) * pot2_1(n/2);
}

unsigned long pot2_2(unsigned n) {
 if (n==0) return 1;
 return 2 * pot2_2(n-1);
}
```

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. La primera, pot2\_1(n), es más eficiente que la otra.
- c. Las dos funciones son equivalentes en cuanto a coste temporal.
- d. La segunda, pot2\_2(n), es más eficiente que la otra.



La respuesta correcta es: Las dos funciones son equivalentes en cuanto a coste temporal.

**Pregunta 6**

Correcta

Puntúa 1,00 sobre 1,00

Si  $f_1(n) \in O(g_1(n))$  y  $f_2(n) \in O(g_2(n))$  entonces...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Las dos anteriores son ciertas.
- c.  $f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$
- d.  $f_1(n) + f_2(n) \in O(\max(g_1(n), g_2(n)))$



La respuesta correcta es: Las dos anteriores son ciertas.

## Pregunta 7

Correcta

Puntúa 1,00 sobre 1,00

Di cuál de estos resultados de coste temporal asintótico es falsa:

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. La ordenación de un vector usando el algoritmo Quicksort requiere en el peor caso un tiempo en  $\Omega(n^2)$ .
- c. La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en  $O(\log n)$ .
- d. La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en  $\Omega(n^2)$ .



La respuesta correcta es: La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en  $\Omega(n^2)$ .

## Pregunta 8

Correcta

Puntúa 1,00 sobre 1,00

¿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?

Seleccione una:

- a. La combinación de las soluciones a los subproblemas.
- b. No contesto (equivalente a no marcar nada).
- c. El número de llamadas recursivas que se hacen.
- d. La división del problema en subproblemas.



La respuesta correcta es: La división del problema en subproblemas.

## Pregunta 9

Correcta

Puntúa 1,00 sobre 1,00

Se pretende obtener la complejidad temporal en el caso más desfavorable de la siguiente función.

```
int exa (vector < int > & v) {
 int i, sum = 0, n = v.size();
 if (n > 0) {
 int j = n;
 while (sum < 100 and j != 0) {
 j = j / 2;
 sum = 0;
 for (i = j; i < n; i++)
 sum += v[i];
 }
 return j;
 }
 else return -1;
}
```

¿Cuál de las siguientes formulaciones expresa mejor dicho coste?

Seleccione una:

- a.  $c_s(n) = \sum_{j=1}^{\log n} \sum_{i=1}^j \left(\frac{1}{2}\right)^i \in O(n \log n)$
- b. No contesto (equivalente a no marcar nada).
- c.  $c_s(n) = \sum_{k=1}^{\log n+1} (n - n/2^k) \in O(n \log n)$  ✓
- d.  $c_s(n) = \sum_{j=0}^{n/2} \left(\frac{1}{2} \sum_{i=j}^n 1\right) \in O(n \log n)$

La respuesta correcta es:  $c_s(n) = \sum_{k=1}^{\log n+1} (n - n/2^k) \in O(n \log n)$

## Pregunta 10

Incorrecta

Puntúa -0,50 sobre 1,00

Tenemos un vector ordenado de tamaño  $n_o$  y un vector desordenado de tamaño  $n_d$  y queremos obtener un vector ordenado con todos los elementos. ¿Qué será más rápido?

Seleccione una:

- a. Depende de si  $n_o > n_d$  o no.
- b. Insertar los elementos del vector desordenado (uno a uno) en el vector ordenado. ✗
- c. No contesto (equivalente a no marcar nada).
- d. Ordenar el desordenado y luego mezclar las listas.

La respuesta correcta es: Ordenar el desordenado y luego mezclar las listas.

**Pregunta 11**

Incorrecta

Puntúa -0,50 sobre 1,00

Con respecto al parámetro  $n$ , ¿Cuál es la complejidad temporal de la siguiente función?

```
void f(unsigned n){
 if(n < 2) return;
 for(int i = 0; i < pow(n,2); i++)
 cout << "*";
 f(n - 2);
}
```

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $\Theta(n^3)$
- c.  $\Theta(n^2 \log n)$
- d.  $\Theta(n^2)$  ✗

La respuesta correcta es:  $\Theta(n^3)$

**Pregunta 12**

Incorrecta

Puntúa -0,50 sobre 1,00

¿Cuál de las siguientes relaciones de recurrencia expresa el número de llamadas recursivas que hace el algoritmo Mergesort?

Seleccione una:

- a.  $T(n) = n + 2T(n/2)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$  ✗
- b.  $T(n) = 1 + 2T(n/2)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$
- c.  $T(n) = n + T(n/2)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es:  $T(n) = 1 + 2T(n/2)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$

[◀ Entrenamiento\\_primer\\_parcial](#)

Ir a...



**Comenzado el** martes, 23 de marzo de 2021, 15:12

**Estado** Finalizado

**Finalizado en** martes, 23 de marzo de 2021, 15:28

**Tiempo empleado** 16 minutos 40 segundos

**Puntos** 8,00/12,00

**Calificación** **6,67** de 10,00 (**67%**)

Pregunta 1

Correcta

Puntúa 1,00 sobre 1,00

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Di cuál de las siguientes afirmaciones es falsa:

Seleccione una:

- a. Si  $(g(n)) \in O(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.
- b. Si  $(g(n)) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria. ✓
- c. Si  $(g(n)) \in O(n)$  la relación de recurrencia representa la complejidad temporal en el mejor caso del algoritmo de búsqueda del k-ésimo elemento más pequeño de un vector (estudiado en clase).
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es: Si  $(g(n)) \in O(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación por inserción binaria.

## Pregunta 2

Incorrecta

Puntúa 0,00 sobre 1,00

Se pretende obtener la complejidad temporal en el caso más desfavorable de la siguiente función.

```
int exa (vector < int > & v) {
 int i, sum = 0, n = v.size();
 if (n > 0) {
 int j = n;
 while (sum < 100 and j != 0) {
 j = j / 2;
 sum = 0;
 for (i = j; i < n; i++)
 sum += v[i];
 }
 return j;
 }
 else return -1;
}
```

¿Cuál de las siguientes formulaciones expresa dicho coste?

Seleccione una:

- a. Las otras dos opciones son ambas ciertas.
- b.  $c_s(n) = \sum_{k=1}^{\log n + 1} (n-n/2^k) \in O(n \log n)$
- c. No contesto (equivalente a no marcar nada). ✗
- d.  $c_s(n) = n \sum_{j=1}^{\log n} \sum_{i=1}^{2^j} \left(\frac{1}{2}\right)^i \in O(n \log n)$

La respuesta correcta es: Las otras dos opciones son ambas ciertas.

## Pregunta 3

Correcta

Puntúa 1,00 sobre 1,00

Con respecto a la complejidad temporal de la siguiente función, ¿cuál de las siguientes afirmaciones es cierta?

```
unsigned long pot2_1(unsigned n) {
 if (n==0)
 return 1;
 if (n%2==0)
 return pot2_1(n/2) * pot2_1(n/2);
 else
 return 2 * pot2_1(n/2) * pot2_1(n/2);
}
```

Seleccione una:

- a. Las otras dos afirmaciones son ambas falsas.
- b. No contesto (equivalente a no marcar nada).
- c. El coste temporal exacto de la función es  $\mathcal{O}(n)$ . ✓
- d. La complejidad temporal en el mejor de los casos es constante.

La respuesta correcta es: El coste temporal exacto de la función es  $\mathcal{O}(n)$ .

**Pregunta 4**

Incorrecta

Puntúa -0,50 sobre 1,00

Con respecto al parámetro  $\backslash(n\backslash)$ , ¿Cuál es la complejidad temporal de la siguiente función?

```
void f(unsigned n){
 if(n < 1) return;
 for(int i = 0; i < n; i++)
 for(int j = 0; j < n; j++)
 for(int k = 0; k < n; k++)
 cout << "*";
 for(int i = 0; i < 8; i++)
 f(n / 2);
}
```

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $\backslash(\Theta(n^3)\backslash)$  ✗
- c.  $\backslash(\Theta(n^2 \log n)\backslash)$
- d.  $\backslash(\Theta(n^3 \log n)\backslash)$

La respuesta correcta es:  $\backslash(\Theta(n^3 \log n)\backslash)$

**Pregunta 5**

Correcta

Puntúa 1,00 sobre 1,00

Tenemos un vector ordenado de tamaño  $\backslash(n_o\backslash)$  y un vector desordenado de tamaño  $\backslash(n_d\backslash)$  y queremos obtener un vector ordenado con todos los elementos. ¿Qué será más rápido?

Seleccione una:

- a. Insertar los elementos del vector desordenado (uno a uno) en el vector ordenado.
- b. Depende de si  $\backslash(n_o > n_d\backslash)$  o no.
- c. No contesto (equivalente a no marcar nada).
- d. Ordenar el desordenado y luego mezclar las listas. ✓

La respuesta correcta es: Ordenar el desordenado y luego mezclar las listas.

## Pregunta 6

Correcta

Puntúa 1,00 sobre 1,00

Sea la siguiente relación de recurrencia:

$T(n) = \left\{ \begin{array}{ll} 1 & \text{si } n \leq 1 \\ 8T(\frac{n}{8}) + g(n) & \text{en otro caso} \end{array} \right.$

Si  $T(n) \in \Theta(n^2)$ , ¿en cuál de estos tres casos nos podemos encontrar?

Seleccione una:

- a.  $g(n)=n^2$  ✓
- b.  $g(n)=n$
- c. No contesto (equivalente a no marcar nada).
- d.  $g(n)=n^3$

La respuesta correcta es:  $g(n)=n^2$

## Pregunta 7

Correcta

Puntúa 1,00 sobre 1,00

Di cuál de estos resultados de coste temporal asintótico es falsa:

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. La búsqueda binaria en un vector ordenado requiere en el peor caso un tiempo en  $\mathcal{O}(\log n)$
- c. La ordenación de un vector usando el algoritmo Quicksort requiere en el peor caso un tiempo en  $\mathcal{O}(n^2)$
- d. La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en  $\mathcal{O}(n^2)$  ✓

La respuesta correcta es: La ordenación de un vector usando el algoritmo Mergesort requiere en el peor caso un tiempo en  $\mathcal{O}(n^2)$ .

## Pregunta 8

Correcta

Puntúa 1,00 sobre 1,00

¿Qué se entiende por tamaño del problema?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. El valor máximo que puede tomar una instancia cualquiera de ese problema.
- c. El número de parámetros que componen el problema.
- d. La cantidad de espacio en memoria que se necesita para codificar una instancia de ese problema. ✓

La respuesta correcta es: La cantidad de espacio en memoria que se necesita para codificar una instancia de ese problema.

## Pregunta 9

Incorrecta

Puntúa -0,50 sobre 1,00

Si  $\{f\} \notin O(g_1)$  y  $\{f\} \in O(g_2)$  entonces NO siempre se cumplirá:

Seleccione una:

- a.  $\{f\} \in \Omega(g_1 + g_2)$
- b.  $\{f\} \in \Omega(\min(g_1, g_2))$  ✗
- c. No contesto (equivalente a no marcar nada).
- d.  $\{f\} \in O(\max(g_1, g_2))$

La respuesta correcta es:  $\{f\} \in \Omega(g_1 + g_2)$

## Pregunta 10

Correcta

Puntúa 1,00 sobre 1,00

De las siguientes tres afirmaciones, una es cierta y dos falsas, o bien una es falsa y dos son ciertas. Marca la que en ese sentido es diferente a las otras dos.

Seleccione una:

- a.  $O(n^2) \subset O(2^{\lfloor \log_2(n) \rfloor}) \subset O(2^n)$  ✓
- b.  $O(2^{\lfloor \log_2(n) \rfloor}) \subseteq O(n^2) \subset O(2^n)$
- c.  $O(4^{\lfloor \log_2(n) \rfloor}) \subseteq O(n^2) \subset O(2^n)$
- d. No contesto (equivalente a no marcar nada).

La respuesta correcta es:  $O(n^2) \subset O(2^{\lfloor \log_2(n) \rfloor}) \subset O(2^n)$

## Pregunta 11

Correcta

Puntúa 1,00 sobre 1,00

Los algoritmos de ordenación Quicksort y Mergesort tienen en común ...

Seleccione una:

- a. ... que ordenan el vector sin usar espacio adicional.
- b. ... que aplican la estrategia de divide y vencerás. ✓
- c. No contesto (equivalente a no marcar nada).
- d. ... que se ejecutan en tiempo  $O(n)$ .

La respuesta correcta es: ... que aplican la estrategia de divide y vencerás.

## Pregunta 12

Correcta

Puntúa 1,00 sobre 1,00

Si  $f(n) \in O(n^3)$ , ¿puede pasar que  $f(n) \in O(n^2)$ ?

Seleccione una:

- a. Sólo para valores bajos de  $n$
- b. Es perfectamente posible, ya que  $O(n^2) \subset O(n^3)$  ✓
- c. No contesto (equivalente a no marcar nada).
- d. No, porque  $n^3 \not\in O(n^2)$

La respuesta correcta es: Es perfectamente posible, ya que  $O(n^2) \subset O(n^3)$

[◀ Entrenamiento\\_primer\\_parcial](#)

Ir a...

Con respecto al parámetro  $n$ , ¿Cuál es la complejidad temporal de la siguiente función?

```
void f(unsigned n) {
 if(n < 2) return;
 for(int i = 0; i < pow(n, 3); i++)
 cout << "*";
 f(n / 2);
}
```

Seleccione una:

- a.  $\Theta(n^3)$
- b. No contesto (equivalente a no marcar nada).
- c.  $\Theta(n^4)$
- d.  $\Theta(n^3 \log n)$



¿Cuál de las siguientes relaciones de recurrencia expresa mejor la **complejidad espacial** del algoritmo Mergesort?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $T(n) = n + T(n - 1)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$
- c.  $T(n) = n + T(n/2)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$
- d.  $T(n) = n + 2T(n/2)$  para  $n > 1$  y  $T(n) = 1$  para  $n \leq 1$  ✓

Los algoritmos de ordenación Quicksort y Mergesort tienen en común ...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... que ordenan el vector sin usar espacio adicional.
- c. ... que se ejecutan en tiempo  $O(n)$ .
- d. ... que aplican la estrategia de divide y vencerás.



Si  $f \notin O(g_1)$  y  $f \in O(g_2)$  entonces siempre se cumplirá:

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $f \notin O(\max(g_1, g_2))$
- c.  $f \in \Omega(g_1 + g_2)$
- d.  $f \in \Omega(\min(g_1, g_2))$

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.

Seleccione una:

- a.  $\Theta(\log_2(n)) = \Theta(\log_3(n))$
- b. No contesto (equivalente a no marcar nada).
- c.  $\log(n^3) \notin \Theta(\log_3(n))$
- d.  $\Theta(\log^2(n)) = \Theta(\log^3(n))$

✗

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?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Sólo si  $d \log d > k \cdot n$  (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.
- d. Sí, ya que el Mergesort es  $O(n \log n)$  y este es  $O(n)$



Las siguientes funciones calculan el valor de la potencia n-ésima de dos. ¿Cuál es más eficiente en cuanto a coste temporal?

```
unsigned long pot2_1(unsigned n) {
 if (n==0) return 1;
 if (n%2==0) return pot2_1(n/2) * pot2_1(n/2);
 else return 2 * pot2_1(n/2) * pot2_1(n/2);
}
```

```
unsigned long pot2_2(unsigned n) {
 if (n==0) return 1;
 return 2 * pot2_2(n-1);
}
```

Seleccione una:

- a. La primera,  $\text{pot2\_1}(n)$ , es más eficiente que la otra. ✗
- b. Las dos funciones son equivalentes en cuanto a coste temporal.
- c. No contesto (equivalente a no marcar nada).
- d. La segunda,  $\text{pot2\_2}(n)$ , es más eficiente que la otra.

¿En qué caso la complejidad temporal de Quicksort es la misma que la del algoritmo de ordenación por inserción?

Seleccione una:

- a. En el caso peor.
- b. No contesto (equivalente a no marcar nada).
- c. En ningún caso.
- d. En el caso mejor.

La complejidad temporal en el mejor de los casos de un algoritmo recursivo...

Seleccione una:

- a. ... es la complejidad temporal de las instancias que están en el caso base.
- b. ... coincide con el valor del caso base de la ecuación de recurrencia que expresa la complejidad temporal del algoritmo.
- c. ... ninguna de las otras dos opciones es cierta.
- d. No contesto (equivalente a no marcar nada).



Se pretende obtener la complejidad temporal en el caso más desfavorable de la siguiente función.

```
int exa (vector<int> & v) {
 int i, sum = 0, n = v.size();
 if (n > 0) {
 int j = n;
 while (sum < 100 and j != 0) {
 j = j / 2;
 sum = 0;
 for (i = j; i < n; i++)
 sum += v[i];
 }
 return j;
 }
 else return -1;
}
```

¿Cuál de las siguientes formulaciones expresa dicho coste?

Seleccione una:

- a. Las otras dos opciones son ambas claras.
- b.  $c_s(n) = n \sum_{j=1}^{\log n} \sum_{i=1}^j \left(\frac{1}{2}\right)^i \in O(n \log n)$
- c.  $c_s(n) = \sum_{k=1}^{\log n+1} (n - n/2^k) \in O(n \log n)$
- d. No contesto (equivalente a no marcar nada). x

La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Dí cuál de las siguientes afirmaciones es falsa:

Seleccione una:

- a. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal en el caso mejor del algoritmo de ordenación Quicksort.
- b. Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación Mergesort.
- c. No contesto (equivalente a no marcar nada).
- d. Si  $g(n) \in \Theta(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

Pertenece  $3n^2 + 3$  a  $O(n^3)$ ?

Seleccione una:

- a. Solo para  $c = 6$  y  $n_0 = 1$
- b. No.
- c. No contesto (equivalente a no marcar nada).
- d. Sí



¡ ¿Qué esquema algorítmico utiliza el algoritmos de ordenación Quicksort?

- a. Divide y Vencerás
- b. Programación Dinámica
- c. Backtracking

2

Ante un problema que presenta una solución recursiva siempre podemos aplicar:

- a. Divide y vencerás
- b. Programación dinámica
- c. Cualquiera de las dos anteriores

3

En cual de los siguientes casos no se puede aplicar el esquema Divide y Vencerás:

- a. Cuando los subproblemas son de tamaños muy diferentes
- b. Cuando el problema no cumple el principio de optimalidad
- c. Se puede aplicar en ambos casos.

4

Dado el algoritmo de búsqueda binaria, supongamos que, en vez de dividir la lista de elementos en dos mitades del mismo tamaño, la dividamos en dos partes de tamaños  $1/3$  y  $2/3$ . El coste de este algoritmo:

- a. Es el mismo que el del original
- b. Es mayor que el del original
- c. Es menor que el del original

5

Si  $n$  es el número de elementos del vector, el coste del algoritmo Mergesort es:

- a.  $O(n^2)$  y  $\Omega(n \log n)$
- b.  $\Theta(n \log n)$
- c.  $\Theta(n^2)$

6

Un problema se puede resolver por Divide y Vencerás siempre que:

- a. Cumpla el principio de optimalidad
- b. Cumpla el teorema de reducción
- c. Ninguna de las anteriores

7

La serie de números de Fibonacci se define de la siguiente forma:

$$fib(n) = \begin{cases} 1 & n \leq 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$$

Para implementar esta función podemos emplear :

- a. Divide y vencerás
- b. Programación dinámica
- c. Cualquiera de las dos anteriores

8

La serie de números de Fibonacci se define de la siguiente forma:

$$fib(n) = \begin{cases} 1 & n \leq 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$$

¿Qué implementación de entre las siguientes supone el menor coste?

- a. Divide y vencerás
- b. Programación dinámica
- c. Ambas tienen el mismo coste asintótico

9

El problema de la mochila, ¿puede solucionarse de forma óptima empleando la estrategia de divide y vencerás?:

- a. Sólo para el caso de la mochila con fraccionamiento
- b. Sólo para el caso de la mochila sin fraccionamiento
- c. Si, se puede aplicar para ambos casos.

10

Para que un problema de optimización se pueda resolver mediante PD es necesario que:

- a. Cumpla el principio de optimalidad
- b. Cumpla el teorema de reducción
- c. Cumpla los dos anteriores

c. Cumpla los dos anteriores

11

Dada una solución recursiva a un problema ¿Cómo podemos evitar la resolución de los mismos subproblemas muchas veces?

- a. Resolver los subproblemas de mayor a menor y guardar su resultado en una tabla, inicializándola con los problemas pequeños.
- b. Resolver los subproblemas de menor a mayor y guardar su resultado en una tabla, inicializándola con los problemas pequeños.
- c. Resolver los subproblemas de mayor a menor y guardar su resultado en una tabla, inicializándola con los problemas más grandes.

12

- Si aplicamos Programación Dinámica a un problema que también tiene solución por divide y vencerás asegurarás que..
- a. El coste temporal se reduce y el espacial aumenta con respecto a la solución por DyV
  - b. El coste temporal aumenta y el espacial se reduce con respecto a la solución por DyV
  - c. Ninguna de las anteriores.

13

¿Cuándo utilizaremos Programación Dinámica en lugar de Divide y Vencerás?

- a. Cuando se incrementa la eficacia
- b. Cuando se incrementa la eficiencia
- c. Cuando se reduce el coste espacial.

14

En programación dinámica, dónde almacenamos los valores de los problemas resueltos?

- a. En un vector unidimensional
- b. En un vector bidimensional
- c. Depende del problema

15

Supongamos el problema de la mochila resuelto mediante Programación Dinámica y particularizado para  $n$  elementos y un peso máximo trasportable de  $P$ . ¿Es necesario calcular valores para toda la matriz auxiliar para obtener el resultado?

a. Si

b. No

c. Depende de los valores de  $n$  y  $P$ .

16

Un problema de optimización cuya solución se puede expresar mediante una secuencia de decisiones cumple el principio de optimalidad si, dada una secuencia óptima:

- a. Existe una subsecuencia de esa solución que corresponde a la solución óptima de su subproblema asociado
- b. Existe al menos una subsecuencia de esa solución que corresponde a la solución óptima de su subproblema asociado
- c. Cualquier subsecuencia de esa solución corresponde a la solución óptima de su subproblema asociado

17

La programación dinámica, para resolver un problema, aplica la estrategia...

- a. Se resuelven los problemas más pequeños y, combinando las soluciones de problemas sucesivamente más grandes hasta llegar al problema original)
- b. Se descompone el problema a resolver en subproblemas más pequeños, que se resuelven independientemente para finalmente combinar las soluciones de los subproblemas para obtener la solución del problema original.
- c. Ninguna de las anteriores

18

¿Qué esquema de programación es el adecuado para resolver el problema del k-ésimo mínimo en un vector?

- a. Programación Dinámica
- b. Divide y Vencerás
- c. Ninguno de los dos

19

Si  $n$  es el número de elementos de un vector. La solución de menor coste al problema de encontrar su  $k$ -ésimo mínimo tiene la siguiente complejidad:

- a.  $\Omega(n)$  y  $O(n \log n)$
- b.  $\Omega(n)$  y  $O(n^2)$
- c. Ninguna de las dos

20

Si  $n$  es el número de elementos de un vector. Podemos encontrar una solución al problema de encontrar su  $k$ -ésimo que esté acotada superiormente por :

- a.  $O(n^3)$
- b.  $O(n)$
- c. Ninguna de las dos

| 21 Dada la solución recursiva al problema de encontrar el k-ésimo mínimo de un vector. Cada llamada recursiva, ¿cuántas nuevas llamadas recursivas genera? |                  |
|------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| a. una o ninguna                                                                                                                                           | b. dos o ninguna |

- La solución al problema de encontrar el  $k$ -ésimo mínimo de un vector pone en práctica la siguiente estrategia:
- a. Ordena totalmente el vector
  - b. Ordena parcialmente el vector
  - c. No ordena ningún elemento del vector

23

¿Qué esquema de programación es el adecuado para resolver el problema de la búsqueda binaria?

- a. Programación Dinámica
- b. Divide y Vencerás
- c. Ninguno de los dos

c. Ninguno de los dos

24

Si  $n$  es el número de elementos de un vector. La solución de menor coste al problema de la búsqueda binaria tiene la siguiente complejidad:

- a.  $\Omega(\log n)$  y  $O(n \log n)$
- b.  $\Theta(n \log n)$
- c.  $\Omega(1)$  y  $O(\log n)$

25

¿Con qué esquema de programación obtenemos algoritmos que calculan la distancia de edición entre dos cadenas?

- a. Programación Dinámica
- b. Divide y vencerás
- c. Ambos

| e. Alumnos |                                                                                                                                                                                                                                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26         | <p>Disponemos de dos cadenas de longitudes m y n. Si resolvemos el problema de la distancia de edición mediante programación dinámica, ¿De qué tamaño debemos definir la matriz que necesitaremos?</p> <p>a. <math>(m-1) \times (n-1)</math></p> <p>b. <math>m \times n</math></p> <p>c. <math>(m+1) \times (n+1)</math></p> |

- 1 El método voraz se emplea en la resolución de problemas de selección y optimización en los que se pretende encontrar:
- a. Una solución que satisfaga unas restricciones y optimice una cierta función objetivo.
  - b. Todas las soluciones que satisfagan unas restricciones.
  - c. La dos respuestas anteriores son correctas.

2

Voraz siempre da solución óptima:

- a. Al problema de la mochila sin fraccionamiento.
- b. Al problema de la mochila con fraccionamiento.
- c. A los dos.

3

En el método Voraz, aunque las decisiones son irreversibles, podemos asegurar que:

- a. Siempre obtendremos la solución óptima.
- b. Siempre obtendremos una solución factible.
- c. Sólo obtendremos la solución óptima para algunos problemas.

4

Dado un problema de optimización y un algoritmo Voraz que lo soluciona, ¿cuándo podemos estar seguros de que la solución obtenida será óptima?:

- a. Cuando demostremos formalmente que el criterio conduce a una solución óptima para cualquier instancia del problema.
- b. Voraz siempre encuentra solución óptima.
- c. En ambos casos. Las dos son correctas

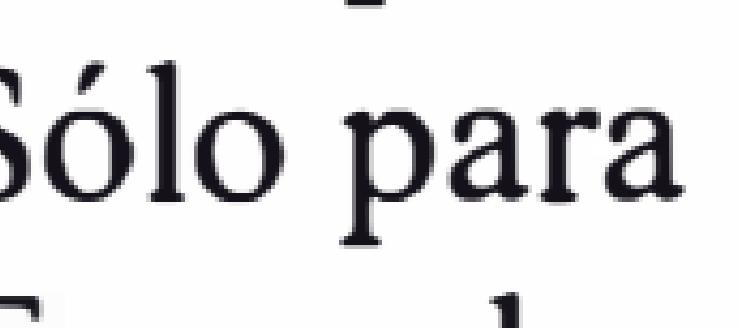
5

Si aplicamos un algoritmo voraz que no nos garantiza la solución óptima sobre un problema entonces...

- a. Obtendremos una solución factible.
- b. Puede que no encuentre ninguna solución aunque ésta exista.
- c. Si el problema tiene solución óptima, el esquema voraz nos garantiza que la encuentra.

6

El problema de la mochila, ¿encuentra su solución óptima empleando la estrategia voraz?:

- a. Sólo para el caso de la mochila con fraccionamiento
-  b. Sólo para el caso de la mochila sin fraccionamiento
- c. En cualquiera de los casos anteriores.

7

Dado un grafo  $G$  que representa las poblaciones de la provincia de Alicante de más de 20.000 habitantes junto con todas las carreteras de conexión entre ellas. Queremos obtener el recorrido que nos permita pasar por todas estas ciudades una única vez y volver al punto de origen recorriendo el mínimo número de kilómetros. Si aplicamos una estrategia voraz sobre este grafo obtendremos...

- a. Una solución factible
- b. La solución óptima
- c. Puede que no encuentre ninguna solución aunque ésta exista.

8

Al aplicar backtracking obtenemos la solución óptima a un problema:

- a. Siempre
- b. En algunos casos
- c. Sólo cuando el problema cumple el principio de Optimalidad.

9

Si aplicamos un esquema backtracking que no nos garantiza la solución óptima sobre un problema entonces.

- a. Obtendremos una solución factible.
- b. Puede que no encuentre ninguna solución aunque ésta exista.
- c. Ninguna de las anteriores.

10

Backtracking es aplicable a problemas de selección y optimización en los que:

a. El espacio de soluciones es un conjunto infinito.

b. El espacio de soluciones es un conjunto finito.

c. En cualquiera de los casos

11

En un problema resuelto por backtracking, el conjunto de valores que pueden tomar las componentes de la tupla solución, ha de ser:

- a. Infinito.
- b. finito
- c. continuo

| 12 Al aplicar vuelta atrás a la solución de problemas, obtenemos algoritmos con costes computacionales: |                   |
|---------------------------------------------------------------------------------------------------------|-------------------|
| a. Polinómicos.                                                                                         | b. Exponenciales. |

13

- Vuelta atrás se emplea en la resolución de problemas de optimización en los que se pretende encontrar:
- a. Una solución que satisfaga unas restricciones y optimice una cierta función objetivo.
  - b. Todas las soluciones que satisfagan unas restricciones.
  - c. La dos respuestas anteriores son correctas.

14

Bactracking procede a obtener la solución a un problema de optimización empleando la siguiente estrategia:

- a. Genera todas las combinaciones de la solución y optimiza la función objetivo.
- b. Genera todas las soluciones factibles y optimiza la función objetivo.
- c. Genera una solución factible empleando un criterio óptimo.

15

Bactracking es una técnicas de resolución general de problemas basada en:

-  a. La búsqueda sistemática de soluciones.
- b. La construcción directa de la solución.
- c. Ninguna de las anteriores

Bactracking genera las soluciones posibles al problema:

- a. Mediante el recorrido en profundidad del árbol que representa el espacio de soluciones.
- b. Mediante el recorrido en anchura del árbol que representa el espacio de soluciones
- c. Ninguna de las anteriores

17

El problema de la mochila, ¿puede solucionarse empleando vuelta atrás?:

- a. Sólo para el caso de la mochila con fraccionamiento
- b. Sólo para el caso de la mochila sin fraccionamiento
- c. Se puede aplicar para ambos casos.

- 18 El problema del viajante de comercio puede resolverse correctamente empleando estos esquemas de programación:
- a. Solo backtracking
  - b. Empleando cualquiera de estos: Voraz y backtracking.
  - c. Sólo programación dinámica.

17

El problema de la mochila, ¿puede solucionarse empleando vuelta atrás?

- a. Sólo para el caso de la mochila con fraccionamiento
-  b. Sólo para el caso de la mochila sin fraccionamiento
- c. Se puede aplicar para ambos casos.

18

El problema del viajante de comercio puede resolverse correctamente empleando estos esquemas de programación:

- a. Solo backtracking
- b. Empleando cualquiera de estos: Voraz y backtracking.
- c. Sólo programación dinámica.

19

El tiempo de ejecución de un algoritmo de ramificación y poda depende de:

- a. La instancia del problema
- b. La función de selección de nodos para su expansión
- c. De ambos

20

Dado un problema resuelto mediante backtracking y mediante ramificación y poda, el coste computacional de la solución por ramificación y poda, en comparación con la de backtracking es:

- a. Igual
- b. Mayor
- c. Menor.

| 21 Cuál de estas afirmaciones es falsa? |                                                                                                                         |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
|                                         | a. Backtracking inspecciona todo el espacio de soluciones de un problema mientras que Ramificación y poda no.           |
|                                         | b. La complejidad en el peor caso de las soluciones Backtracking y ramificación y poda a un mismo problema es la misma. |
|                                         | c. Para un mismo problema, ramificación y poda explora siempre un número menor o igual que backtracking.                |

## **EL PROBLEMA DE LA ASIGNACIÓN DE TURNOS.**

Estamos al comienzo del curso y los alumnos quieren buscar compañero para formar un grupo de prácticas. Para solucionar este problema se propone que elijan a varias personas y les asignen una prioridad. El número de alumnos es  $N$ . Se dispone una matriz cuadrada  $M$  con  $N$  filas en la que cada alumno escribió, en su fila correspondiente, un número entero (entre  $-1$  y  $N-1$ ) indicando dicha prioridad (un valor  $-1$  indica que no quiere o no puede estar con la persona de la columna correspondiente,  $0$  indica indiferencia y, cuanto más alto es, mayor es la preferencia por esa persona) . Ningún alumno puede formar grupo consigo mismo.

Se pretende encontrar una solución para satisfacer el número máximo de alumnos según su orden de preferencia. Suponiendo que la matriz  $M$  ya existe, diseñar un algoritmo que resuelva el problema de forma óptima

22

El problema de la asignación de turnos tiene solución óptima voraz aplicando la siguiente estrategia:

- a. Seleccionamos los alumnos en orden descendente de preferencia respetando las restricciones de cabida de cada turno.
- b. Seleccionamos los alumnos en orden ascendente de preferencia respetando las restricciones de cabida de cada turno
- c. El problema no tiene solución óptima voraz

23

El problema de la asignación de turnos tiene solución óptima empleando:

- a. Bactracking
- b. Voraz
- c. Ambos

24

El problema de la asignación de turnos tiene solución...

- a. Optima mediante bactracking
- b. Aproximada (sub-óptima) mediante voraz
- c. Ambas.

24

El problema de la asignación de turnos tiene solución...

- a. Optima mediante backtracking
- b. Aproximada (sub-óptima) mediante voraz
- c. Ambas.

25

Dada la solución recursiva mediante vuelta atrás al problema de turnos ¿cuántas nuevas llamadas recursivas genera cada llamada recursiva?

- a. una o dos
- b. una o ninguna
- c. ninguna de las anteriores

26

El problema de la asignación de turnos resuelto mediante backtracking tiene una complejidad:

-  a. Exponencial
- b. Polinómica
- c. Ninguna de la dos

Pregunta 6

Correcto

Puntaje como 1.00

 Marcar pregunta

¿Cuál de estos tres problemas de optimización no tiene o no se le conoce una solución voraz óptima?

Seleccione una:

- a. El **árbol de cobertura de coste mínimo de un grafo conexo**
- b. El **problema de la mochila discreta o sin fraccionamiento** ✓
- c. El **problema de la mochila continua o con fraccionamiento**

Pregunta 8

Correcta

Punt. 3 como 1.00

Marcar pregunta

Los algoritmos de programación dinámica hacen uso

Seleccione una:

- a de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes uno a uno
- b de que se puede ahorrar cálculos guardando resultados anteriores en un almacen ✓
- c de una estrategia óptima consistente en examinar todas las soluciones posibles

Pregunta 10

Correcto

Puntuación 1.00

Puntaje Pregunta

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos.

Seleccione una

- a. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.
- b. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.
- c. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.

**2012-13\_ANALISIS Y DISEÑO DE ALGORITMOS\_34018**Usted se ha identificado como [Victor Garcia Ruiz \(Salir\)](#)
[Página Principal](#) ► [Mis cursos](#) ► [ADA\\_34018](#) ► [Segundo examen parcial de ADA](#) ► [Segundo parcial](#)

**Navegación por el cuestionario**

[Mostrar una página cada vez](#)

[Finalizar revisión](#)

|                        |                                  |
|------------------------|----------------------------------|
| <b>Comenzado el</b>    | lunes, 6 de mayo de 2013, 11:21  |
| <b>Completado el</b>   | lunes, 6 de mayo de 2013, 12:08  |
| <b>Tiempo empleado</b> | 46 minutos 14 segundos           |
| <b>Puntos</b>          | 4,50/12,00                       |
| <b>Calificación</b>    | 3,75 de un máximo de 10,00 (38%) |

**Pregunta 1** Se pretende implementar mediante programación dinámica iterativa la función recursiva:

Correcta  
Puntúa 1,00 sobre 1,00

Demarcar

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A  
 b. int A[][]  
 c. int A[] ✓

La respuesta correcta es: int A[]

**Pregunta 2**

Incorrecta  
Puntúa -0,50 sobre 1,00

Marcar pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo voraz.  
 b. ... un algoritmo del estilo de *divide y vencerás*. X  
 c. ... un algoritmo de programación dinámica.

La respuesta correcta es: ... un algoritmo de programación dinámica.

**Pregunta 3**

Correcta  
Puntúa 1,00 sobre 1,00

Demarcar

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Seleccione una:

- a. ... tiene un coste temporal asintótico exponencial con respecto al número de objetos.  
 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. ✓

La respuesta correcta es: ... tiene la restricción de que los valores tienen que ser enteros positivos.

**Pregunta 4**

Incorrecta  
Puntúa -0,50 sobre 1,00

Demarcar

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo de programación dinámica. X  
 b. un algoritmo voraz.

- c. un algoritmo divide y vencerás.

La respuesta correcta es: un algoritmo voraz.

**Pregunta 5**

Incorrecta

Puntúa -0,50 sobre  
1,00

 Demarcar

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n=0 \\ \sum_{k=0}^{n-1} T(k) & n>0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $T(n) \in \Theta(n^2)$  X
- b.  $T(n) \in \Theta(n!)$
- c.  $T(n) \in \Theta(2^n)$

La respuesta correcta es:  $T(n) \in \Theta(2^n)$

**Pregunta 6**

Correcta

Puntúa 1,00 sobre  
1,00

 Demarcar

En el método voraz ...

Seleccione una:

- a. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓
- c. ... siempre se encuentra solución pero puede que no sea la óptima.

La respuesta correcta es: ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.

**Pregunta 7**

Correcta

Puntúa 1,00 sobre  
1,00

 Marcar  
pregunta

¿ Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores ?

Seleccione una:

- a. Ya no se garantizaría la solución óptima pero sí una factible.
- b. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar.
- c. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna. ✓

La respuesta correcta es: La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna.

**Pregunta 8**

Incorrecta

Puntúa -0,50 sobre  
1,00

 Demarcar

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- b. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar. X
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar.

La respuesta correcta es: Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar.

**Pregunta 9**

Correcta

Puntúa 1,00 sobre  
1,00

Demarcar

Dado un problema de optimización, el método voraz ...

Seleccione una:

- a. ... siempre obtiene la solución óptima.
- b. ... garantiza la solución óptima sólo para determinados problemas. ✓
- c. ... siempre obtiene una solución factible.

La respuesta correcta es: ... garantiza la solución óptima sólo para determinados problemas.

**Pregunta 10**

Correcta

Puntúa 1,00 sobre  
1,00

Demarcar

Cuando se calculan los coeficientes binomiales usando la recursión

$$\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}, \text{ con } \binom{n}{0} = \binom{n}{n} = 1,$$
 qué problema se da y cómo se puede resolver?

Seleccione una:

- a. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.
- b. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica. ✓
- c. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.

La respuesta correcta es: Se repiten muchos cálculos y ello se puede evitar usando programación dinámica.

**Pregunta 11**

Incorrecta

Puntúa -0,50 sobre  
1,00 Marcar  
pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(x^2)$  ✗
- b.  $O(1)$
- c.  $O(x)$

La respuesta correcta es:  $O(1)$ **Pregunta 12**

Correcta

Puntúa 1,00 sobre  
1,00

Demarcar

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2

centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Hacer una evaluación exhaustiva ``de fuerza bruta'' de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .
- b. Hacer una evaluación exhaustiva ``de fuerza bruta'' de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ . ✓
- c. Es posible evitar hacer la evaluación exhaustiva ``de fuerza bruta'' guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.

La respuesta correcta es: Hacer una evaluación exhaustiva ``de fuerza bruta'' de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ .[Finalizar revisión](#)

Usted se ha identificado como Victor Garcia Ruiz (Salir)

ADA\_34018

**2012-13\_ANALISIS Y DISEÑO DE ALGORITMOS\_34018**

[Página Principal](#) ► Mis cursos ► [ADA\\_34018](#) ► Segundo examen parcial de ADA ► [Segundo parcial](#)

**Navegación por el cuestionario**[Mostrar una página cada vez](#)[Finalizar revisión](#)

|                        |                                  |
|------------------------|----------------------------------|
| <b>Comenzado el</b>    | lunes, 6 de mayo de 2013, 11:22  |
| <b>Complegado el</b>   | lunes, 6 de mayo de 2013, 11:58  |
| <b>Tiempo empleado</b> | 36 minutos 25 segundos           |
| <b>Puntos</b>          | 4,00/12,00                       |
| <b>Calificación</b>    | 3,33 de un máximo de 10,00 (33%) |

**Pregunta 1**

Correcta

Puntúa 1,00 sobre  
1,00

Marcar pregunta

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Seleccione una:

- a. ... las decisiones tomadas nunca se reconsideran. ✓
- b. ... con antelación, las posibles decisiones se ordenan de mejor a peor.
- c. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.

La respuesta correcta es: ... las decisiones tomadas nunca se reconsideran.

**Pregunta 2**

Incorrecta

Puntúa -0,50 sobre  
1,00

Marcar pregunta

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n=0 \\ \sum_{k=0}^{n-1} T(k) & n>0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $T(n) \in \Theta(n!)$  ✗
- b.  $T(n) \in \Theta(n^2)$
- c.  $T(n) \in \Theta(2^n)$

La respuesta correcta es:  $T(n) \in \Theta(2^n)$ **Pregunta 3**

Correcta

Puntúa 1,00 sobre  
1,00

Marcar pregunta

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...

Seleccione una:

- a. ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño 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 pocas veces un número relativamente grande de subproblemas distintos.

La respuesta correcta es: ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos.

**Pregunta 4**

Correcta

Puntúa 1,00 sobre  
1,00

Marcar pregunta

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Seleccione una:

- a. Ninguna de las otras dos opciones es cierta. ✓
- b. ... la solución óptima está garantizada.
- c. ... al menos una solución factible está garantizada.

La respuesta correcta es: Ninguna de las otras dos opciones es cierta.

**Pregunta 5**

Correcta

Puntúa 1,00 sobre  
1,00 Marcar  
pregunta

¿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?

Seleccione una:

- a. Programación dinámica. ✓
- b. Las dos estrategias citadas serían similares en cuanto a eficiencia.
- c. La estrategia voraz.

La respuesta correcta es: Programación dinámica.

**Pregunta 6**

Incorrecta

Puntúa -0,50 sobre  
1,00 Marcar  
pregunta

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- 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 el mínimo de monedas posibles.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.

La respuesta correcta es: El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.

**Pregunta 7**

Incorrecta

Puntúa -0,50 sobre  
1,00 Marcar  
pregunta

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz. ✗
- b. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar.

La respuesta correcta es: Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar.

**Pregunta 8**

Sin contestar

Puntúa como 1,00

 Marcar  
pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(y^2)$
- b.  $O(y)$
- c.  $O(1)$

La respuesta correcta es:  $O(y)$

**Pregunta 9**

Incorrecta

Puntúa -0,50 sobre  
1,00 Marcar  
pregunta

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y la mochila continua.
- b. El fontanero diligente y el problema del cambio. ✗
- c. El fontanero diligente y la asignación de tareas.

La respuesta correcta es: El fontanero diligente y la mochila continua.

**Pregunta 10**

Sin contestar

Puntúa como 1,00

 Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A
- b. int A[]
- c. int A[][]

La respuesta correcta es: int A[]

**Pregunta 11**

Correcta

Puntúa 1,00 sobre 1,00

 Marcar pregunta

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo divide y vencerás.
- b. un algoritmo voraz. ✓
- c. un algoritmo de programación dinámica.

La respuesta correcta es: un algoritmo voraz.

**Pregunta 12**

Correcta

Puntúa 1,00 sobre 1,00

 Marcar pregunta

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Seleccione una:

- a. El método voraz.
- b. Programación dinámica. ✓
- c. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.

La respuesta correcta es: Programación dinámica.

[Finalizar revisión](#)

Usted se ha identificado como [Alberto Martinez Lopez \(Salir\)](#)

ADA\_34018

## SEGUNDO PARCIAL

**Pregunta 6** Correcta  
Puntúa como 1,00

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

a. El árbol de cobertura de coste mínimo de un grafo conexo.

b. El problema de la mochila discreta o sin fraccionamiento. ✓

c. El problema de la mochila continua o con fraccionamiento.

**Pregunta 8** Correcta  
Puntúa como 1,00

Los algoritmos de programación dinámica hacen uso ...

Seleccione una:

a. ... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.

b. ... de que se puede ahorrar cálculos guardando resultados anteriores en un almacén. ✓

c. ... de una estrategia trivial consistente en examinar todas las soluciones posibles.

**Pregunta 10** Correcta  
Puntúa como 1,00

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

a. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.

b. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓

c. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.

**Pregunta 1** Correcta  
Puntúa como 1,00

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

a. un algoritmo voraz. ✓

b. un algoritmo divide y vencerás.

c. un algoritmo de programación dinámica.

**Pregunta 2** Correcta  
Puntúa como 1,00

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...

Seleccione una:

a. ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.

b. ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos. ✓

c. El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.

**Pregunta 3** En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Correcta

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓
- b. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

**Pregunta 4**

Correcta

Puntúa como 1,00

Marcar pregunta

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n = 0 \\ \sum_{k=0}^{n-1} T(k) & n > 0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $T(n) \in \Theta(n^2)$
- b.  $T(n) \in \Theta(n!)$
- c.  $T(n) \in \Theta(2^n)$  ✓

**Pregunta 5**

Correcta

Puntúa como 1,00

Marcar pregunta

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Seleccione una:

- a. El método voraz.
- b. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
- c. Programación dinámica. ✓

**Pregunta 7**

En el método voraz ...

Correcta

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- a. ... siempre se encuentra solución pero puede que no sea la óptima.
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓
- c. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

**Pregunta 8**

Correcta

Puntúa como 1.00

Marcar pregunta

¿ Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores ?

Seleccione una:

- a. Ya no se garantizaría la solución óptima pero sí una factible.
- b. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible.
- c. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna. ✓

**Pregunta 9**

Correcta

Puntúa como 1.00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A
- b. int A[] ✓
- c. int A[][]

**Pregunta 8**

Correcta

Puntúa como 1.00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A[]
- b. int A
- c. int A[][] ✓

**Pregunta 12**

Correcta

Puntúa como 1,00

Marcar pregunta

¿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?

Seleccione una:

- a. La estrategia voraz.
- b. Programación dinámica. ✓
- c. Las dos estrategias citadas serían similares en cuanto a eficiencia.

**Pregunta 2**

Correcta

Puntúa como 1,00

Marcar pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo voraz.
- b. ... un algoritmo de programación dinámica. ✓
- c. ... un algoritmo del estilo de divide y vencerás.

**Pregunta 11**

Correcta

Puntúa como 1,00

Marcar pregunta

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- 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 el mínimo de monedas posibles.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓

**Pregunta 5**

Correcta

Puntúa como 1,00

Marcar pregunta

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Seleccione una:

- a. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.
- b. ... tiene un coste temporal asintótico exponencial con respecto al número de objetos
- c. ... tiene la restricción de que los valores tienen que ser enteros positivos. ✓

**Pregunta 6**

Correcta

Puntúa como 1,00

Marcar pregunta

Cuando se calculan los coeficientes binomiales usando la recursión  $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$ , con  $\binom{n}{0} = \binom{n}{n} = 1$ , qué problema se da y cómo resolver?

Seleccione una:

El problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo.
- b. ... no se puede resolver en general con una estrategia voraz.
- c. ... se puede resolver siempre con una estrategia voraz. ✓

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Seleccione una:

- a. ... al menos una solución factible está garantizada.
- b. Ninguna de las otras dos opciones es cierta. ✓
- c. ... la solución óptima está garantizada.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x) { // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$
- b.  $O(y^2)$
- c.  $O(y)$  ✓

El valor que se obtiene con el método voraz para el problema de la mochila discreta es

...

Seleccione una:

- a. ... una cota inferior para el valor óptimo que a veces puede ser igual a este. ✓
- b. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
- c. ... una cota superior para el valor óptimo.

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de menor peso.
- b. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓
- c. Meter primero los elementos de mayor valor.

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Seleccione una:

- a. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.
- b. ... con antelación, las posibles decisiones se ordenan de mejor a peor.
- c. ... las decisiones tomadas nunca se reconsideran. ✓

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...

Seleccione una:

- 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. ✓

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo de programación dinámica. ✓
- 

Pregunta 4

Correcta

Puntúa como 1.00

Marcar pregunta

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

- a. El problema de la mochila discreta o sin fraccionamiento. ✓
- b. El árbol de cobertura de coste mínimo de un grafo conexo.
- c. El problema de la mochila continua o con fraccionamiento.

**Pregunta 6**

Correcta

Puntúa como 1,00

Marcar pregunta

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Seleccione una:

- a. ... la solución óptima está garantizada.
- b. Ninguna de las otras dos opciones es cierta. ✓
- c. ... al menos una solución factible está garantizada.

**Pregunta 10**

Correcta

Puntúa como 1,00

Marcar pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo del estilo de *divide y vencerás*.
- b. ... un algoritmo de programación dinámica. ✓
- c. ... un algoritmo voraz.

**Pregunta 3**

Correcta

Puntúa como 1,00

Marcar pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo del estilo de *divide y vencerás*.
- b. ... un algoritmo voraz.
- c. ... un algoritmo de programación dinámica. ✓

**Pregunta 7**

Correcta

Puntúa como 1,00

Marcar pregunta

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓
- b. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

**Pregunta 3** ¿ Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores ?

Marcar pregunta

Seleccione una:

- a. Ya no se garantizaría la solución óptima pero sí una factible.
- b. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna. ✓
- c. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible.

**Pregunta 7**

Correcta

Puntúa como 1,00

Marcar pregunta

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Seleccione una:

- a. Programación dinámica. ✓
- b. El método voraz.
- c. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.

**Pregunta 2**

Correcta

Puntúa como 1,00

Marcar pregunta

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .
- b. Es posible evitar hacer la evaluación exhaustiva "de fuerza bruta" guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.
- c. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ . ✓

**Pregunta 3**

Correcta

Puntúa como 1,00

Marcar pregunta

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- b. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

**Pregunta 10**

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Correcta

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- a. Ninguna de las otras dos opciones es cierta. ✓
- b. ... al menos una solución factible está garantizada.
- c. ... la solución óptima está garantizada.

**Pregunta 10**

Correcta

Puntúa como 1.00

Marcar pregunta

¿Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores?

Seleccione una:

- a. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible.
- b. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna. ✓
- c. Ya no se garantizaría la solución óptima pero sí una factible.

**Pregunta 11**

Correcta

Puntúa como 1.00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(x^2)$
- b.  $O(1)$  ✓
- c.  $O(x)$

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A,2+B);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(y^2)$
- b.  $O(1)$
- c.  $O(y)$  ✓

**Pregunta 12**

Correcta

Puntúa como 1.00

Marcar pregunta

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Seleccione una:

- a. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.
- b. ... tiene un coste temporal asintótico exponencial con respecto al número de objetos.
- c. ... tiene la restricción de que los valores tienen que ser enteros positivos. ✓

**Pregunta 5**

Correcta

Puntúa como 1.00

Marcar pregunta

La programación dinámica...

Seleccione una:

- a. ... en algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos.
- b. Las otras dos opciones son ciertas. ✓
- c. ... normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores.

**Pregunta 9**

Correcta

Puntúa como 1.00

Marcar pregunta

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y la mochila continua. ✓
- b. El fontanero diligente y la asignación de tareas.
- c. El fontanero diligente y el problema del cambio.

**Pregunta 1**

Un algoritmo recursivo basado en el esquema *divide y vencerás* ...

Correcta

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- a. Las demás opciones son verdaderas.
- b. ... nunca tendrá una complejidad exponencial.
- c. ... será más eficiente cuanto más equitativa sea la división en subproblemas.

**Pregunta 8**

Indicad cuál de estas tres expresiones es cierta:

Correcta

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- a.  $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
- b.  $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$
- c.  $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$

**Pregunta 3**

Dado un problema de optimización, el método voraz ...

Correcta

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- a. ... siempre obtiene una solución factible.
- b. ... siempre obtiene la solución óptima.
- c. ... garantiza la solución óptima sólo para determinados problemas.

**Pregunta 5**

Correcta

Puntúa como 1,00

Marcar pregunta

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarse en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo voraz.
- b. un algoritmo de programación dinámica.
- c. un algoritmo divide y vencerás.

**Pregunta 8**

Correcta

Puntúa como 1.00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$  ✓
- b.  $O(x^2)$
- c.  $O(x)$

**Pregunta 12**

Correcta

Puntúa como 1.00

Marcar pregunta

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Dí cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Es posible evitar hacer la evaluación exhaustiva "de fuerza bruta" guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.
- b. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ . ✓
- c. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .

**Pregunta 2**

Correcta

Puntúa como 1.00

Marcar pregunta

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...

Seleccione una:

- a. ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos. ✓
- b. ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.
- c. El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.

**Pregunta 9**

Correcta

Puntúa como 1.00

Marcar pregunta

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de menor peso.
- b. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓
- c. Meter primero los elementos de mayor valor.

**Pregunta 2**

Correcta

Puntúa como 1.00

Marcar pregunta

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- b. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

**Pregunta 4**

Correcta

Puntúa como 1.00

Marcar pregunta

En el método voraz

Seleccione una:

- a. ... siempre se encuentra solución pero puede que no sea la óptima.
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓
- c. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

**Pregunta 1**

Correcta

Puntúa como 1.00

 Marcar pregunta

Un informático quiere subir a una montaña y para ello decide que tías cada paso, el siguiente debe tomarse en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿Qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo voraz. ✓
- b. un algoritmo divide y vencerás.
- c. un algoritmo de programación dinámica.

## Pregunta 2

Conecta

Puntúa como 1.00

 **Martar pregunta**

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...

Seleccione una:

- a. ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.
- b. ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos. ✓
- c. El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.

Pregunta 3

Correcta

Puntúa como 1.00

 Marcar pregunta

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓
- b. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

## Pregunta 4

Correcta

Puntúa como 1,00

Marcar pregunta

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n = 0 \\ \sum_{k=0}^{n-1} T(k) & n > 0 \end{cases}$$

¿Cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $T(n) \in \Theta(n^2)$
- b.  $T(n) \in \Theta(n!)$
- c.  $T(n) \in \Theta(2^n)$  ✓

**Pregunta 5**

Correcta

Puntúa como 1,00

 Marcar pregunta

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Seleccione una:

- a. El método voraz.
- b. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
- c. Programación dinámica. ✓

**Pregunta 7**

En el método voraz ...

Correcta

Puntúa como 1,00

 Marcar pregunta

Seleccione una:

- a. ... siempre se encuentra solución pero puede que no sea la óptima.
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓
- c. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

**Pregunta 8**

Correcta

Puntúa como 1.00

 Marcar pregunta

¿ Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores ?

Seleccione una:

- a. Ya no se garantizaría la solución óptima pero sí una factible.
- b. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible.
- c. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna. ✓

**Pregunta 9**

Corrección

Puntúa como 1.00

 Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

 a. int A b. int A[] ✓ c. int A[][]

Pregunta 8

Correcta

Puntúa como 1,00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A[]
- b. int A
- c. int A[][] ✓

## Pregunta 12

Conecta

Puntúa como 1.00

 Marcar pregunta

¿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?

Seleccione una:

- a. La estrategia voraz.
- b. Programación dinámica. ✓
- c. Las dos estrategias citadas serían similares en cuanto a eficiencia.

Pregunta 2

Correcta

Puntuación 1.00

 Marcar pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo voraz.
- b. ... un algoritmo de programación dinámica. ✓
- c. ... un algoritmo del estilo de divide y vencerás.

## Pregunta 11

Correcta

Puntúa como 1.00

 Marcar pregunta

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- 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 el mínimo de monedas posibles.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓

Pregunta 5

Correcto

Puntaje como 1.00

 **Marcar pregunta**

## La solución de programación dinámica iterativa del problema de la mochila discreta

Seleccione una:

a. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.

b. ... tiene un coste temporal asintótico exponencial con respecto al número de objetos

c. ... tiene la restricción de que los valores tienen que ser enteros positivos. ✓

El problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo.
- b. ... no se puede resolver en general con una estrategia voraz.
- c. ... se puede resolver siempre con una estrategia voraz. ✓

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Seleccione una:

- a. ... al menos una solución factible está garantizada.
- b. Ninguna de las otras dos opciones es cierta. ✓
- c. ... la solución óptima está garantizada.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x) { // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$
- b.  $O(y^2)$
- c.  $O(y)$  ✓

El valor que se obtiene con el método voraz para el problema de la mochila discreta es

...

Seleccione una:

- a. ... una cota inferior para el valor óptimo que a veces puede ser igual a este. ✓
- b. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
- c. ... una cota superior para el valor óptimo.

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de menor peso.
- b. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓
- c. Meter primero los elementos de mayor valor.

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Seleccione una:

- a. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.
- b. ... con antelación, las posibles decisiones se ordenan de mejor a peor.
- c. ... las decisiones tomadas nunca se reconsideran. ✓

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...

Seleccione una:

- 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. ✓

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:



- a. ... un algoritmo de programación dinámica. ✓

Pregunta 4

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Correctas

Puntúa como 1.00

 Marcar pregunta

Seleccione una:

- a. El problema de la mochila discreta o sin fraccionamiento. ✓
- b. El árbol de cobertura de coste mínimo de un grafo conexo.
- c. El problema de la mochila continua o con fraccionamiento.

Pregunta 6

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Correcto

Puntúa como 1.00

 Marcar pregunta

Seleccione una:

- a. ... la solución óptima está garantizada.
- b. Ninguna de las otras dos opciones es cierta. ✓
- c. ... al menos una solución factible está garantizada.

**Pregunta 10**

Correcta

Puntúa como 1.00

 Marcar pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños y, por otro lado, los subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en...

Selección una:

- a ... un algoritmo del estilo de donde y vencerás.
- b ... un algoritmo de programación dinámica. ✓
- c ... un algoritmo voraz.

Pregunta 3

Correcta

Puntúa como 1,00

 Marcar pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo del estilo de *divide y vencerás*.
- b. ... un algoritmo voraz.
- c. ... un algoritmo de programación dinámica. ✓

Pregunta 7

Correcta

Puntúa como 1,00

 Marcar pregunta

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓
- b. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

Pregunta 3

Correccla

Puntúa como 1,00

 Marcar pregunta

¿ Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores ?

Seleccione una:

- a. Ya no se garantizaría la solución óptima pero sí una factible.
- b. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna. ✓
- c. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible.

## Pregunta 7

Correcta

Puntúa como 1.00

 Marcar pregunta

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Seleccione una:

- a. Programación dinámica. ✓
- b. El método voraz.
- c. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.

**Pregunta 2**  
Correcta

Puntaje como 1,00

 Marcar  
pregunta

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .
- b. Es posible evitar hacer la evaluación exhaustiva "de fuerza bruta" guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.
- c. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ . 

### Pregunta 3

Comida

Puntúa como 1.00

Marcar

Pregunta

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- b. Para reducir la complejidad temporal en la toma de cada decisión de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓
- c. Para reducir la complejidad temporal en la toma de cada decisión de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

**Pregunta 10**

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Correcta

Puntaje como 1,00

 Marcar pregunta

Seleccione una:

- a. Ninguna de las otras dos opciones es cierta. ✓
- b. ... al menos una solución factible está garantizada.
- c. ... la solución óptima está garantizada.

Pregunta 10

Correcto

Puntaje como 1.00

 Marcar pregunta

¿ Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores ?

Seleccione una:

- a. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible.
- b. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a una solución alguna. ✓
- c. Ya no se garantizaría la solución óptima pero sí una factible.

Pregunta 11

Correcta

Puntaje como 1.00

 Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

a.  $O(x^2)$

b.  $O(1)$  ✓

c.  $O(x)$

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A,2+B);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(y^2)$
- b.  $O(1)$
- c.  $O(y)$  ✓

Pregunta 12

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Correida

Puntuación 1.00

 Marcar pregunta

Seleccione una:

- a. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.
- b. ... tiene un coste temporal asintótico exponencial con respecto al número de objetos.
- c. ... tiene la restricción de que los valores tienen que ser enteros positivos. ✓

**Pregunta 5**

Correcto

Puntúa como 1.00

 Marcar pregunta

## La programación dinámica...

Seleccione una:

- a. ... en algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos.
- b. Las otras dos opciones son ciertas. ✓
- c. ... normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores.

Pregunta 9

Correcta

Puntúa como 1.00

Marcar

pregunta

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y la mochila continua. ✓
- b. El fontanero diligente y la asignación de tareas.
- c. El fontanero diligente y el problema del cambio.

Pregunta 1

Correcta

Puntúa como 1,00

 Marcar pregunta

Un algoritmo recursivo basado en el esquema *divide y vencerás* . . .

Seleccione una:

- a. Las demás opciones son verdaderas.
- b. ... nunca tendrá una complejidad exponencial.
- c. ... será más eficiente cuanto más equitativa sea la división en subproblemas. ✓

**Pregunta 8**

Indicad cuál de estas tres expresiones es cierta:

Correcta

Puntúa como 1,00

 Marcar  
pregunta

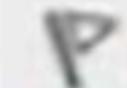
Seleccione una:

- a.  $O(n^2) \subset O(2^{\log(n)}) \subset O(2^n)$
- b.  $O(2^{\log(n)}) \subset O(n^2) \subset O(2^n)$  ✓
- c.  $O(n^2) \subset O(2^{\log(n)}) \subseteq O(2^n)$

Pregunta 3

Correcto

Puntúa como 1.00

 Marcar  
puntuación

Dado un problema de optimización, el método voraz ...

Seleccione una:

- a. ... siempre obtiene una solución factible.
- b. ... siempre obtiene la solución óptima.
- c. ... garantiza la solución óptima sólo para determinados problemas. ✓

**Pregunta 5**  
Correcta

Puntúa como 1.00

 Marcar pregunta

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo voraz. ✓
- b. un algoritmo de programación dinámica.
- c. un algoritmo divide y vencerás.

## Pregunta 8

Correcta

Puntuación 1.00

 Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$  ✓
- b.  $O(x^2)$
- c.  $O(x)$

## Pregunta 12

Correcta

Puntúa como 1.00

 Marcar pregunta

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Es posible evitar hacer la evaluación exhaustiva "de fuerza bruta" guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.
- b. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ . ✓
- c. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .

Pregunta 2

Correcta

Puntúa como 1.00

 Marcar pregunta

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...

Seleccione una:

- a. ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos. ✓
- b. ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.
- c. El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.

Pregunta 9

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Correcta

Puntúa como 1.00

 Marcar  
pregunta

Seleccione una:

- a. Meter primero los elementos de menor peso.
- b. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓
- c. Meter primero los elementos de mayor valor.

**Pregunta 2**

Correctas

Puntúa como 1.00

Marca,  
pregunta

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- b. Para reducir la complejidad temporal en la toma de cada decisión de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓
- c. Para reducir la complejidad temporal en la toma de cada decisión de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

Pregunta 4

Correcto

Puntuación 1.00

 **Marcar pregunta**

En el método voraz

Seleccione una:

- a. ... siempre se encuentra solución pero puede que no sea la óptima.
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓
- c. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

Pregunta 1

Correcta

Puntuación 1.00

 Marcar pregunta

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarse en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo voraz. ✓
- b. un algoritmo divide y vencerás.
- c. un algoritmo de programación dinámica.

Pregunta 2

Correcta

Puntúa como 1.00

 Marcar pregunta

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...

Seleccione una:

- a. ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.
- b. ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos. ✓
- c. El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.

### Pregunta 3

Correcta

Puntúa como 1.00

 Marcar pregunta

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Para introducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$  donde  $n$  es el número de objetos a considerar. ✓
- b. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

Pregunta 4

Correcta

Puntúa como 1.00

Marcar  
pregunta

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n = 0 \\ \sum_{k=0}^{n-1} T(k) & n > 0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

a.  $T(n) \in \Theta(n^2)$

b.  $T(n) \in \Theta(n!)$

c.  $T(n) \in \Theta(2^n)$  ✓

Pregunta 5

Correcta

Puntúa como 1.00

 Marcar pregunta

Quando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Seleccione una:

- a. El método voraz.
- b. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
- c. Programación dinámica ✓

**Pregunta 6**

Sin contestar

Puntúa como 1.00

 Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A,2+B);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

a.  $O(y)$

b.  $O(1)$

c.  $O(y^2)$

Pregunta 7

En el método voraz ...

Correcta

Puntuación 1.00

 Marcar  
puntuación

Seleccione una:

- a. ... siempre se encuentra solución pero puede que no sea la óptima.
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓
- c. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

Pregunta 8

Correcta

Puntúa como 1.00

 Marcar  
pregunta

¿ Cómo se veía afectada la solución VRaz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores ?

Seleccione una:

- a. Ya no se garantizaría la solución óptima pero si una factible.
- b. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible.
- c. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna. ✓

**Pregunta 9**

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

Correcta

Puntuó como 1.00

Marcar pregunta

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

a. int A

b. int A[] ✓

c. int A[][]

**Pregunta 10**

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Correcta

Puntúa como 1.00

 Marcar pregunta

Seleccione una:

- a. Ninguna de las otras dos opciones es cierta. ✓
- b. ... al menos una solución factible está garantizada.
- c. ... la solución óptima está garantizada.

Pregunta 11

Correcta

Puntuación 1,00

 Marcar pregunta

De los problemas siguientes, indicar cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema de cortar un lulo 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 el mínimo de monedas posibles.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al contenido de los pesos de los objetos y de sus valores. ✓

Pregunta 12

Correcta

Puntúa como 1.00

 Marcar pregunta

¿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?

Seleccione una:

- a. La estrategia voraz.
- b. Programación dinámica. ✓
- c. Las dos estrategias citadas serían similares en cuanto a eficiencia.

**Pregunta 1** De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Corrección

Puntúa como 1.00

 Marcar

Pregunta

Seleccione una:

- 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 el mínimo de monedas posibles.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓

Pregunta 2

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Correcta

Puntúa como 1.00

 Marcar pregunta

Seleccione una:

a. ... las decisiones tomadas nunca se reconsideran. ✓

b. ... con antelación, las posibles decisiones se ordenan de mejor a peor.

c. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.

Pregunta 3

Correcta

Puntúa como 1.00

 Marcar pregunta

Dado un problema de optimización, el método voraz ...

Seleccione una:

- a. ... siempre obtiene una solución factible.
- b. ... siempre obtiene la solución óptima.
- c. ... garantiza la solución óptima sólo para determinados problemas. ✓

**Pregunta 4**  
Correcta

Puntúa como 1.00

 Marcar pregunta

¿ Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores ?

Seleccione una:

- a. Ya no se garantizaría la solución óptima pero sí una factible.
- b. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a una solución alguna. ✓
- c. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible.

Pregunta 5

Correcta

Puntúa como 1,00

 Marcar pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo de programación dinámica. ✓
- b. ... un algoritmo del estilo de *divide y vencerás*.
- c. ... un algoritmo voraz.

## Pregunta 6

Correcto

Puntúa como 1.00

Marcar pregunta

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n = 0 \\ \sum_{k=0}^{n-1} T(k) & n > 0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $T(n) \in \Theta(n^2)$
- b.  $T(n) \in \Theta(2^n)$  ✓
- c.  $T(n) \in \Theta(n!)$

Pregunta 1

Sin contestar

Puntuación: 0,00

 Marcar pregunta

## La programación dinámica...

Seleccione una:

- a. ... en algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos.
- b. Las otras dos opciones son ciertas.
- c. ... normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores.

**Pregunta 2**

Correcta

Puntuación: 1,00

 **Marcar pregunta**

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Seleccione una:

- a. ... la solución óptima está garantizada.
- b. ... al menos una solución factible está garantizada.
- c. Ninguna de las otras dos opciones es cierta. ✓

Pregunta 3

Conecta

Puntaje como 1.00

 Marcar pregunta

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Seleccione una:

- a. Programación dinámica. ✓
- b. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
- c. El método voraz.

Pregunta 4

Correcta

Puntúa como 1.00

 Marca pregunta

El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Seleccione una:

- a. ... una cota superior para el valor óptimo.
- b. ... una cota inferior para el valor óptimo que a veces puede ser igual a este. ✓
- c. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.

Pregunta 5  
Conecta

Puntúa como 1.00

Marcar pregunta

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿Qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo voraz. ✓
- b. un algoritmo de programación dinámica.
- c. un algoritmo divide y vencerás.

Pregunta 6  
Correcta

Puntúa como 1.00

 Marca pregunta

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y el problema del cambio.
- b. La mochila continua y la asignación de tareas.
- c. La mochila discreta y la asignación de tareas. ✓

Pregunta 7

Correcta

Puntuación 1,00

 Marcar pregunta

¿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?

Seleccione una:

- a. La estrategia voraz.
- b. Las dos estrategias citadas serían similares en cuanto a eficiencia.
- c. Programación dinámica. ✓

Pregunta 8

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$  ✓
- b.  $O(x^2)$
- c.  $O(x)$

Pregunta 9

Correcta

Puntuación como 1.00

Marcar pregunta

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n = 0 \\ \sum_{k=0}^{n-1} T(k) & n > 0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $T(n) \in \Theta(n^2)$
- b.  $T(n) \in \Theta(n!)$
- c.  $T(n) \in \Theta(2^n)$  ✓

Pregunta 10

Correcta

Puntúa como 1.00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A[] ✓
- b. int A
- c. int A[][]

Pregunta 11

Correcta

Puntuación: 1.00

 Marcar pregunta

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- 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 el mínimo de monedas posibles.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓

Pregunta 12

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Correcto

Puntúa como 1.00

 Marcar pregunta

Seleccione una:

- a. ... las decisiones tomadas nunca se reconsideran. ✓
- b. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.
- c. ... con antelación, las posibles decisiones se ordenan de mejor a peor.

Pregunta 7

Correcta

Puntúa como 1.00

 Marcar pregunta

¿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?

Seleccione una:

- a. Las dos estrategias citadas serían similares en cuanto a eficiencia.
- b. La estrategia voraz.
- c. Programación dinámica. ✓

Pregunta 8

Correcta

Puntaje como 1.00

 Marcar pregunta

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿Qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo voraz. ✓
- b. un algoritmo divide y vencerás.
- c. un algoritmo de programación dinámica.

Pregunta 9

Correcta

Puntúa como 1,00

 Marcar pregunta

El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Seleccione una:

- a. ... una cota inferior para el valor óptimo que a veces puede ser igual a este. ✓
- b. ... una cota superior para el valor óptimo.
- c. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.

Pregunta 10

Correcta

Puntúa como 1.00

 Marcar

Pregunta

Los algoritmos de programación dinámica hacen uso ...

Seleccione una:

- a. ... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.
- b. ... de que se puede ahorrar cálculos guardando resultados anteriores en un almacén. ✓
- c. ... de una estrategia trivial consistente en examinar todas las soluciones posibles.

## Pregunta 11

Sin contestar

Puntúa como 1.00

 Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A,2+B);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$
- b.  $O(y^2)$
- c.  $O(y)$

Pregunta 12

Corrección

Puntúa como 1.00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

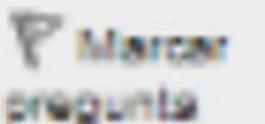
a. int A[][] ✓

b. int A[]

c. int A

Preguntas 1  
Correctas

Punto 1.00 sobre  
1.00



Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarse 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. ¿que tipo de algoritmo esta usando nuestro informático?

Seleccione una:

- a. un algoritmo divide y vencerás.
- b. un algoritmo de programación dinámica.
- c. un algoritmo voraz. ✓

La respuesta correcta es un algoritmo voraz.

Pregunta 2  
Coreada

Puntuación 1.00 sobre  
1.00

Marcar  
pregunta

Cuando se calculan los coeficientes binomiales usando la recursión  $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$ , con  $\binom{n}{0} = \binom{n}{n} = 1$ , qué problema se da y cómo se puede resolver?

Seleccione una:

- a. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia traz.
- b. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica. ✓
- c. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.

La respuesta correcta es: Se repiten muchos cálculos y ello se puede evitar usando programación dinámica.

**Preguntas 3**

Comedia

Puntuación 1 000 sobre

1.00

 Marcar pregunta

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos.

Seleccione una:

- a. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓
- b. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.
- c. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.

Pregunta 4

Sin contestar

Puntaje como 1,00

Marcar pregunta

En la solución al problema de la mochila continua ¿por que es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar.
- b. Para reducir la complejidad temporal en la toma de cada decisión de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.
- c. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz

La respuesta correcta es: Para reducir la complejidad temporal en la toma de cada decisión de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar.

Pregunta 5

Correcta

Puntaje 1.00 sobre  
1.00

▼ Marcar pregunta

La etiología de los algoritmos voraces se basa en el hecho de que ...

Selecione una:

- a. ... las decisiones tomadas nunca se reconsideran. ✓
- b. ... con antelación, las posibles decisiones se ordenan de mejor a peor.
- c. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.

La respuesta correcta es: ... las decisiones tomadas nunca se reconsideran.

### Pregunta 6

Correcto

Puntaje 1,00 sobre  
1,00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor estructura para el almacen?

Seleccione una:

- A. int A
- B. int A[][] ✓
- C. int A[]

La respuesta correcta es: int A[][]

Pregunta 7

Correcta

Puntuación: 1,00 sobre

1,00

Puntaje global:

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que...

Seleccione una:

- a. ... en la solución ingenua se resuelven 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. ✓

La respuesta correcta es: ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos.

**Pregunta 8**

Sin contestar

Puntaje como 1.00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x) { // exponente y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Selección una:

- a.  $O(y)$
- b.  $O(y^2)$
- c.  $O(1)$

La respuesta correcta es:  $O(y)$

**Pregunta 9**

Sin contestar

Puntúa como 1.00

Markar pregunta

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n=0 \\ \sum_{k=0}^{n-1} T(k) & n>0 \end{cases}$$

¿Cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $T(n) \in \Theta(2^n)$
- b.  $T(n) \in \Theta(n^2)$
- c.  $T(n) \in \Theta(n!)$

La respuesta correcta es:  $T(n) \in \Theta(2^n)$

**Pregunta 10**

Correcta

Puntaje 1,00 sobre  
1,00

Marcar pregunta

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Seleccione una:

- a. Ninguna de las otras dos opciones es cierta.
- b. ... al menos una solución factible está garantizada.
- c. la solución óptima está garantizada.

La respuesta correcta es: Ninguna de las otras dos opciones es cierta.

Pregunta 11

Correcto

Punto 1.00 score  
1.00

Finalizar pregunta

¿Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores?

Seleccione una:

- a. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar.
- b. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna. ✓
- c. Ya no se garantizaría la solución óptima pero sí una factible.

La respuesta correcta es: La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna.

Pregunta 12

Correcta

Puntaje 1,00 sobre  
1,00

▼ Marcar pregunta

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Selección rápida:

- a. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
- b. Programación dinámica. ✓
- c. El método voraz.

La respuesta correcta es: Programación dinámica.

Pregunta 1

Respuesta  
guardada

Puntuación 1,00

▼ Marcar  
pregunta

El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Seleccione una:

- a. ... una cota superior para el valor óptimo.
- b. ... una cota inferior para el valor óptimo, pero que nunca coincide con este
- c. ... una cota inferior para el valor óptimo que a veces puede ser igual a este

## Pregunta 2

Sin responder aún

Puntúa como 1,00

 Marcar pregunta

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Es posible evitar hacer la evaluación exhaustiva "de fuerza bruta" guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.
- b. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .
- c. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ .

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.

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

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.

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.

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.

## Pregunta 12

Respuesta  
guardada

Puntúa como 1,00

 Marcar  
pregunta

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?



Seleccione una:

- a. El árbol de cobertura de coste mínimo de un grafo conexo.
- b. El problema de la mochila continua o con fraccionamiento.
- c. El problema de la mochila discreta o sin fraccionamiento.

## Pregunta 12

Respuesta  
guardada

Puntúa como 1,00

 Marcar  
pregunta

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima? 

Seleccione una:

- a. El árbol de cobertura de coste mínimo de un grafo conexo.
- b. El problema de la mochila continua o con fraccionamiento.
- c. El problema de la mochila discreta o sin fraccionamiento.

Pregunta 11

Respuesta  
guardada

Puntúa como 1.00

Marcar  
pregunta

Dado un problema de optimización, el método voraz ...

Seleccione una:

- a. ... siempre obtiene una solución factible.
- b. ... garantiza la solución óptima sólo para determinados problemas.
- c. ... siempre obtiene la solución óptima.

**Pregunta 10**

Respuesta  
guardada

Puntúa como 1,00

Marcar  
pregunta

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) apor:tada por el método voraz?

Seleccione una:

- a. El fontanero diligente y el problema del cambio.
- b. El fontanero diligente y la mochila continua.
- c. El fontanero diligente y la asignación de tareas.

Pregunta 9

Sin responder aún

Puntúa como 1,00

▼ Marcar  
pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

C a. int A

C b. int A[][]

C c. int A[ ]

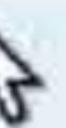
Pregunta 8

Respuesta  
guardada

Puntúa como 1,00

Marca  
pregunta

Los algoritmos de programación dinámica hacen uso ...



Seleccione una:

- a. ... de que se puede ahorrar cálculos guardando resultados anteriores en un almacén.
- b. ... de una estrategia trivial consistente en examinar todas las soluciones posibles.
- c. ... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.

## Pregunta 7

Respuesta  
guardada

Puntúa como 1,00

Marcar  
pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$
- b.  $O(x)$
- c.  $O(x^2)$

### Pregunta 6

Respuesta  
guardada

Puntúa como 1,00

 Marcar  
pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ... 

Seleccione una:

- a. ... un algoritmo del estilo de *divide y vencerás*.
- b. ... un algoritmo de programación dinámica.
- c. ... un algoritmo voraz.

Pregunta 5

Respuesta  
guardada

Puntúa como 1,00

 Marcar  
pregunta

En el método voraz ...

Seleccione una:

- a. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.
- b. ... siempre se encuentra solución pero puede que no sea la óptima.
- c. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.

**Pregunta 4**

Sin responder aún

Puntúa como 1.00

 Marcar pregunta

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de mayor valor específico o valor por unidad de peso
- b. Meter primero los elementos de mayor valor.
- c. Meter primero los elementos de menor peso

Pregunta 3

Respuesta  
guardada

Puntúa como 1,00

Marcar  
pregunta

Cuando se calculan los coeficientes binomiales usando la recursión

$\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$ , con  $\binom{n}{0} = \binom{n}{n} = 1$ . qué problema se da y cómo se puede resolver?

Seleccione una:

- a. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.
- b. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.
- c. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica.

Resolución recursiva para la programación dinámica.

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A[][] X
- b. int A[]
- c. int A

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- 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 el mínimo de monedas posibles.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacen?

Seleccione una:

- a. int A
- b. int A[][]
- c. int A[]

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓
- b. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.
- c. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.

La respuesta correcta es: El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(y)$
- b.  $O(y^2)$
- c.  $O(1)$

La respuesta correcta es:  $O(y)$

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$
- b.  $O(x)$
- c.  $O(x^2)$

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓
- b. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.
- c. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.

La respuesta correcta es: El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacen?

Seleccione una:

- a. int A
- b. int A[][]
- c. int A[]

El problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. se puede resolver siempre con una estrategia voraz.
- b. ... no se puede resolver en general con una estrategia voraz.
- c. sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo.

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y el problema del cambio.
- b. El fontanero diligente y la mochila continua.
- c. El fontanero diligente y la asignación de tareas.

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 XXXXXX?

```
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)  $p, r, n-i$

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.

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. ¿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.

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.

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 trazo. ¿Qué debería ir en lugar de XXXXXXXX?

```
void fill(price m[]) {
 for (index i=0; i<n; i++) m[i]=-1;
}

price cutrod(length s[], 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(s, m, p)};
 }
 m[n]=q;
 return q;
}
```

- (a)  $n-m[n], m, p$
- (b)  $n-i, m, p$
- (c)  $n, m[n]-1, p$

## Pregunta 12

Incorrecta

Puntaje -0,50 sobre  
1,00

 Marcar pregunta

El problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido y ponderado ...

Seleccione una:

- a. ... no se puede resolver en general con una estrategia voraz.
- b. sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo. 
- c. ... se puede resolver siempre con una estrategia voraz.

La respuesta correcta es: ... se puede resolver siempre con una estrategia voraz.

Pregunta 11

Correcta

Puntuación 1.00 sobre  
1.00

 Marcar pregunta

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de menor peso.
- b. Meter primero los elementos de mayor valor.
- c. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓

Pregunta 10

Correcta

Puntuación 1,00 sobre  
1,00

Marcar pregunta

¿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?

Seleccione una:

- a. Las dos estrategias citadas serían similares en cuanto a eficiencia.
- b. Programación dinámica. ✓
- c. La estrategia voraz.

La respuesta correcta es: Programación dinámica.

Pregunta 10

Correcta

Puntuación 1,00 sobre  
1,00

Marcar pregunta

¿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?

Seleccione una:

- a. Las dos estrategias citadas serían similares en cuanto a eficiencia.
- b. Programación dinámica. ✓
- c. La estrategia voraz.

La respuesta correcta es: Programación dinámica.

Pregunta 9

Incorrecta

Puntuación -0,50 sobre  
1,00

Marcar pregunta

En el método voraz ...

Seleccione una:

- a. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.
- b. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables. X
- c. ... siempre se encuentra solución pero puede que no sea la óptima.

La respuesta correcta es: ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.

Pregunta 8

Sin contestar

Pueda como 1.00

 Marcar pregunta

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .
- b. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ .
- c. Es posible evitar hacer la evaluación exhaustiva "de fuerza bruta" guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.

La respuesta correcta es: Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ .

Pregunta 8

Sin contestar

Pueda como 1.00

 Marcar pregunta

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .
- b. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ .
- c. Es posible evitar hacer la evaluación exhaustiva "de fuerza bruta" guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.

La respuesta correcta es: Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ .

Pregunta 7

Correcta

Puntuación 1.00 sobre  
1.00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

a.  $O(1)$

b.  $O(x^2)$

c.  $O(x)$  ✓

La respuesta correcta es:  $O(x)$

Pregunta 6

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Correcta

Puntaje 1,00 sobre  
1,00

Marcar pregunta

Seleccione una:

- a. El árbol de cobertura de coste mínimo de un grafo conexo.
- b. El problema de la mochila continua o con fraccionamiento.
- c. El problema de la mochila discreta o sin fraccionamiento. ✓

La respuesta correcta es: El problema de la mochila discreta o sin fraccionamiento.

Pregunta 5

Correcto

Puntuación 1.00 sobre

1.00

 Marcar pregunta

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...

Seleccione una:

- a. El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.
- b. ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.
- c. ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos. ✓

La respuesta correcta es: ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos.

**Pregunta 4**

Correcta

Puntúa 1,00 sobre  
1,00

 Marcar  
pregunta

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Seleccione una:

- a. Programación dinámica. ✓
- b. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
- c. El método voraz.

La respuesta correcta es: Programación dinámica.

### Pregunta 3

Correcta

Puntúa 1,00 sobre  
1,00

Marcar  
pregunta

Se pretende implementar mediante programación dinámica recursiva la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float a = 0.0;
 if (v1[y] <= x)
 a = v2[y] + f(x-v1[y], y-1);
 float b = f(x, y-1);
 return min(a,2+b);
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. unsigned A[]
- b. unsigned A
- c. unsigned A[][] ✓

## Pregunta 2

Correcta

Puntúa 1.00 sobre  
1.00

 Marcar  
pregunta

Dado un problema de optimización, el método voraz ...

Seleccione una:

- a. ... siempre obtiene la solución óptima.
- b. ... garantiza la solución óptima sólo para determinados problemas. ✓
- c. ... siempre obtiene una solución factible.

### Pregunta 1

El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Correcta

Puntúa 1,00 sobre  
1,00

 Marcar  
pregunta

Seleccione una:

- a. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
- b. ... una cota inferior para el valor óptimo que a veces puede ser igual a este. ✓
- c. ... una cota superior para el valor óptimo.

La respuesta correcta es: ... una cota inferior para el valor óptimo que a veces puede ser igual a este.

Pregunta 1  
Incorrectas

Puntúa como 1,00

Marcar  
pregunta

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ . X
- b. Es posible evitar hacer la evaluación exhaustiva "de fuerza bruta" guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente
- c. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ .

## Pregunta 2

Contesta

Puntúa como 1.00

 Marcar  
pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ..

Seleccione una:

a. ... un algoritmo voraz.

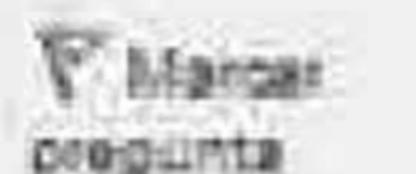
b. ... un algoritmo de programación dinámica ✓

c. ... un algoritmo del estilo de divide y vencerás.

### Pregunta 3

Comenzar

Puntúa como 1.00



¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce una solución voraz óptima?

Seleccione una:

- a El problema de la mochila continua o con fraccionamiento
- b El problema de la mochila discreta o sin fraccionamiento ✓
- c. El árbol de cobertura de coste mínimo de un grafo conexo

Pregunta 4

Correcta

Puntaje como 1.00

 Marcar pregunta

## En el método voraz

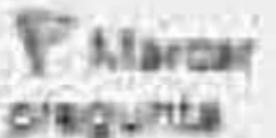
Seleccione una

- a es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar ✓
- b siempre se encuentra solución pero puede que no sea la óptima
- c el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables

**Pregunta 5**

Correcta

Puntuación 1.00



## La solución de programación dinámica iterativa del problema de la mochila discreta

Seleccione una

- a .. calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva
- b .. tiene un coste temporal asintótico exponencial con respecto al número de objetos.
- c .. tiene la restricción de que los valores tienen que ser enteros positivos ✓

Pregunta 6

Cometas

Puntaje como 1.00

 Mejorar  
preguntas

Cuando se calculan los coeficientes binomiales usando la recursión  $\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r}$ , con  $\binom{n}{0} = \binom{n}{n} = 1$ , qué problema se da y cómo se puede resolver?

Seleccione una:

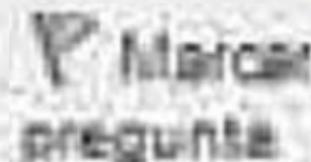
- a La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica
- b Se repiten muchos cálculos y ello se puede evitar usando programación dinámica ✓
- c Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz

## Pregunta 7

Dado un problema de optimización, el método voraz

Correcto

Puntúa como 1.00



Seleccione una.

- a. siempre obtiene la solución óptima
- b. Ninguna de las otras dos opciones es cierta ✓
- c. siempre obtiene una solución factible

Pregunta 8

Correcta

Puntúa como 1.00



Preguntas

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz
- b Para reducir la complejidad temporal en la toma de cada decisión de  $O(n^2)$  a  $O(n \log n)$  donde  $n$  es el número de objetos a considerar
- c Para reducir la complejidad temporal en la toma de cada decisión de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar

Pregunta 9

Correctas

Puntúa como 1.00

 Marcar

Preguntas

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y la asignación de tareas.
- b. El fontanero diligente y el problema del cambio.
- c. El fontanero diligente y la mochila continua. ✓

Pregunta 10

Correcto

Puntaje como 1.00

 Marcar  
pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

a.  $O(x^2)$

b.  $O(x)$

c.  $O(1)$  ✓

### Pregunta 11

Sin contestar

Puntúa como 1.00

Marcar pregunta

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n = 0 \\ \sum_{k=0}^{n-1} T(k) & n > 0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $T(n) \in \Theta(2^n)$
- b.  $T(n) \in \Theta(n^2)$
- c.  $T(n) \in \Theta(n!)$

Pregunta 12

Correcta

Puntúa como 1,00

 Marca pregunta

Se pretende implementar mediante programación dinámica recursiva la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float a = 0.0;
 if (v1[y] <= x)
 a = v2[y] + f(x-v1[y], y-1);
 float b = f(x, y-1);
 return min(a,2+b);
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. unsigned A[]
- b. unsigned A[][] ✓
- c. unsigned A

Cuando se calculan los coeficientes binomiales usando la recursión

$\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$ , con  $\binom{n}{0} = \binom{n}{n} = 1$ , qué problema se da y cómo se puede resolver?

Seleccione una:

- a. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.
- b. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica.
- c. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.
- b. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓

El problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo.
- b. ... no se puede resolver en general con una estrategia voraz.
- c. ... se puede resolver siempre con una estrategia voraz. ✓

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$
- b.  $O(y^2)$
- c.  $O(y)$  ✓

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Seleccione una:

- a. ... al menos una solución factible está garantizada.
- b. Ninguna de las otras dos opciones es cierta. ✓
- c. ... la solución óptima está garantizada.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A[ ] [ ]
- b. int A
- c. int A[ ] X

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo de programación dinámica.
- b. un algoritmo voraz. ✓
- c. un algoritmo divide y vencerás.

El valor que se obtiene con el método voraz para el problema de la mochila discreta es

...

Seleccione una:

- a. ... una cota inferior para el valor óptimo que a veces puede ser igual a este. ✓
- b. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
- c. ... una cota superior para el valor óptimo.

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de menor peso.
- b. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓
- c. Meter primero los elementos de mayor valor.

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.
- b. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓

Cuando se calculan los coeficientes binomiales usando la recursión  
$$\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$$
, con  $\binom{n}{0} = \binom{n}{n} = 1$ , qué problema se da y cómo se puede resolver?

Seleccione una:

- a. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.
- b. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica.
- c. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución

---

alternativa basada en ...

Seleccione una:

- a. ... un algoritmo de programación dinámica. ✓
- b. ... un algoritmo del estilo de *divide y vencerás*.
- c. ... un algoritmo voraz.

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...

Seleccione una:

- 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. ✓

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Seleccione una:

- a. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.
- b. ... con antelación, las posibles decisiones se ordenan de mejor a peor.
- c. ... las decisiones tomadas nunca se reconsideran. ✓

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de menor peso.
- b. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓
- c. Meter primero los elementos de mayor valor.

El valor que se obtiene con el método voraz para el problema de la mochila discreta es

...

Seleccione una:



a. ... una cota inferior para el valor óptimo que a veces puede ser igual a este. ✓



b. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.



c. ... una cota superior para el valor óptimo.

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo de programación dinámica.
- b. un algoritmo voraz. ✓
- c. un algoritmo divide y vencerás.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A[][]
- b. int A
- c. int A[] X

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Seleccione una:

- a. ... al menos una solución factible está garantizada.
- b. Ninguna de las otras dos opciones es cierta. ✓
- c. ... la solución óptima está garantizada.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$
- b.  $O(y^2)$
- c.  $O(y)$  ✓

El problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo.
- b. ... no se puede resolver en general con una estrategia voraz.
- c. ... se puede resolver siempre con una estrategia voraz. ✓

Cuando se calculan los coeficientes binomiales usando la recursión  $\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r-1}$ , con  $\binom{0}{0} = \binom{n}{n} = 1$ , qué problema se da y cómo se puede resolver?

Seleccione una:

- a. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica.
- b. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.
- c. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica. X

Dado un problema de optimización, el método voraz ...

Seleccione una:

- a. ... siempre obtiene una solución factible.
- b. ... garantiza la solución óptima sólo para determinados problemas. ✓
- c. ... siempre obtiene la solución óptima.

Los algoritmos de programación dinámica hacen uso ...

Seleccione una:

- a. ... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.
- b. ... de una estrategia trivial consistente en examinar todas las soluciones posibles.
- c. ... de que se puede ahorrar cálculos guardando resultados anteriores en un almacén. ✓

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

- a. El árbol de cobertura de coste mínimo de un grafo conexo.
- b. El problema de la mochila discreta o sin fraccionamiento. ✓
- c. El problema de la mochila continua o con fraccionamiento.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A, 2+B);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(y^2)$
- b.  $O(1)$
- c.  $O(y)$  ✓

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- 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 el mínimo de monedas posibles.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. La mochila continua y la asignación de tareas.
- b. El fontanero diligente y el problema del cambio.
- c. La mochila discreta y la asignación de tareas.

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar. X
- b. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar.

Se pretende implementar mediante programación dinámica recursiva la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float a = 0.0;
 if (v1[y] <= x)
 a = v2[y] + f(x-v1[y], y-1);
 float b = f(x, y-1);
 return min(a,2+b);
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. unsigned A[]
- b. unsigned A
- c. unsigned A[][] ✓

En el método voraz ...

Seleccione una:

- a. ... siempre se encuentra solución pero puede que no sea la óptima. X
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.
- c. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo de programación dinámica. ✓
- b. ... un algoritmo voraz.
- c. ... un algoritmo del estilo de divide y vencerás.

Seleccione una:

- a. ... un algoritmo de programación dinámica.
- b. ... un algoritmo voraz.
- c. ... un algoritmo del estilo de divide y vencerás.

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de mayor valor.
- b. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓
- c. Meter primero los elementos de menor peso

**Pregunta 12**

Sin contestar

Puntúa como 1,00

¿Para qué se utiliza el TAD "Union-find" en el algoritmo de Kruskal?

Seleccione una:

- a. Para comprobar si un arco forma ciclos.
- b. Para comprobar si un vértice ya ha sido visitado.
- c. Para comprobar si dos vértices son equivalentes.

**Pregunta 11**

Correcta

Puntúa 1,00 sobre  
1,00

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n = 0 \\ \sum_{k=0}^{n-1} T(k) & n > 0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

a.  $T(n) \in \Theta(2^n)$



b.  $T(n) \in \Theta(n^2)$

c.  $T(n) \in \Theta(n!)$

**Pregunta 10**

Correcta

Puntúa 1,00 sobre  
1,00

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y la asignación de tareas.
- b. El fontanero diligente y la mochila continua. ✓
- c. El fontanero diligente y el problema del cambio.

**Pregunta 9**

Correcta

Puntúa 1,00 sobre  
1,00

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar.  

- b. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.
- c. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.

**Pregunta 8**

Correcta

Puntúa 1,00 sobre

1,00

En el método voraz ...

Seleccione una:

- a. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓
- b. ... siempre se encuentra solución pero puede que no sea la óptima.
- c. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

### Pregunta 7

Correcta

Puntúa 1,00 sobre  
1,00

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo divide y vencerás.
- b. un algoritmo voraz. ✓
- c. un algoritmo de programación dinámica.

## Pregunta 6

Correcta

Puntúa 1,00 sobre

1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(x^2)$
- b.  $O(x)$
- c.  $O(1)$



**Pregunta 5**

Correcta

Puntúa 1,00 sobre  
1,00

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓
- b. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.
- c. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.

Pregunta 4

Correcta

Puntúa 1,00 sobre  
1,00

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...

Seleccione una:

- a. ... en la solución ingenua se resuelve pocas veces un número relativamente grande de subproblemas distintos.
- b. ... en la solución ingenua se resuelve muchas veces un número relativamente pequeño de subproblemas distintos. ✓
- c. El número de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programación dinámica.

### Pregunta 3

Correcta

Puntúa 1,00 sobre  
1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A
- b. int A[] ✓
- c. int A[][]

## Pregunta 2

Correcta

Puntúa 1,00 sobre  
1,00

Cuando se calculan los coeficientes binomiales usando la recursión

$\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$ , con  $\binom{n}{0} = \binom{n}{n} = 1$ , qué problema se da y cómo se puede resolver?

Seleccione una:

- a. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.
- b. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica. ✓
- c. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.

**Pregunta 1**

Correcta

Puntúa 1,00 sobre  
1,00

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Seleccione una:

- a. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
- b. El método voraz.
- c. Programación dinámica. ✓

Un tubo de  $74$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:



- a. Hacer una evaluación exhaustiva ``de fuerza bruta'' de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$



- b. Hacer una evaluación exhaustiva ``de fuerza bruta'' de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$  ✓



- c. Es posible evitar hacer la evaluación exhaustiva ``de fuerza bruta'' guardando, para cada posible longitud  $j$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$  **X**
- b.  $O(y^2)$
- c.  $O(y)$

Los algoritmos de programación dinámica hacen uso ...

Seleccione una:



a. ... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.



b. ... de una estrategia trivial consistente en examinar todas las soluciones posibles.



c. ... de que se puede ahorrar cálculos guardando resultados anteriores en un almacén. ✓

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y la asignación de tareas.
- b. El fontanero diligente y la mochila continua. ✓
- c. El fontanero diligente y el problema del cambio.

Dado un problema de optimización, el método voraz ...

Seleccione una:



a. ... siempre obtiene la solución óptima.



b. Ninguna de las otras dos opciones es cierta.



c. ... siempre obtiene una solución factible.

$$\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$$

Cuando se calculan los coeficientes binomiales usando la recursión

$$\binom{n}{0} = \binom{n}{n} = 1$$

, con , qué problema se da y cómo se puede resolver?

Seleccione una:

- a. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.
- b. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica. ✓
- c. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n=0 \\ \sum_{k=0}^{n-1} T(k) & n>0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $T(n) \in \Theta(n^2)$
- b.  $T(n) \in \Theta(n!)$
- c.  $T(n) \in \Theta(2^n)$  ✓

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo voraz.
- b. ... un algoritmo de programación dinámica. ✓
- c. ... un algoritmo del estilo de *divide y vencerás*.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A
- b. int A[] ✓
- c. int A[][]

El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Seleccione una:

- a. ... una cota inferior para el valor óptimo que a veces puede ser igual a este. ✓
- b. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
- c. ... una cota superior para el valor óptimo.

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

- a. El problema de la mochila continua o con fraccionamiento.
- b. El árbol de cobertura de coste mínimo de un grafo conexo.
- c. El problema de la mochila discreta o sin fraccionamiento. ✓

En el método voraz ...

Seleccione una:



a. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.



b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓



c. ... siempre se encuentra solución pero puede que no sea la óptima.

Pregunta 12

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Correcto

Puntuación 1,00

 Marcar pregunta

Seleccione una:

- a. ... tiene la restricción de que los valores tienen que ser enteros positivos. ✓
- b. ... tiene un coste temporal asintótico exponencial con respecto al número de objetos.
- c. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.

Pregunta 11

Correcto

Puntúa como 1.00

 Marcar  
preguntas

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓
- b. Meter primero los elementos de menor peso.
- c. Meter primero los elementos de mayor valor.

Pregunta 10

Correcto

Puntúa como 1,00

 Marcar pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeño; subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo del estilo de *divide y vencerás*.
- b. ... un algoritmo de programación dinámica. ✓
- c. ... un algoritmo voraz.

**Pregunta 9**

Incorrectas

Puntúa como 1.00

 Marcar

Pregunta

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. La mochila discreta y la asignación de tareas.
- b. El fontanero diligente y el problema del cambio. 
- c. La mochila continua y la asignación de tareas.

**Pregunta 8**

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

Corrección

Puntúa como 1.00

 Marcar pregunta

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$
- b.  $O(y)$  ✓
- c.  $O(y^2)$

Pregunta 7

Correcta

Puntúa como 1.00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A[] ✓
- b. int A[][]
- c. int A

**Pregunta 6**

Correcta

Puntaje como 1.00

 Marcar pregunta

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Seleccione una:

- a. .... la solución óptima está garantizada.
- b. Ninguna de las otras dos opciones es cierta. ✓
- c. .... al menos una solución factible está garantizada.

Pregunta 5

Correcto

Puntúa como 1.00

 Marcar pregunta

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Seleccione una:

- a. ... las decisiones tomadas nunca se reconsideran. ✓
- b. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.
- c. ... con antelación, las posibles decisiones se ordenan de mejor a peor.

Pregunta 4

Correcta

Puntúa como 1.00

 Marcar pregunta

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

- a. El problema de la mochila discreta o sin fraccionamiento. ✓
- b. El árbol de cobertura de coste mínimo de un grafo conexo.
- c. El problema de la mochila continua o con fraccionamiento.

**Pregunta 3**

Correcta

Puntúa como 1.00

 Marcar pregunta

¿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?

Seleccione una:

- a. Las dos estrategias citadas serían similares en cuanto a eficiencia.
- b. Programación dinámica. ✓
- c. La estrategia voraz.

**Pregunta 2**

Correcta

Puntúa como 1,00

Marcar

pregunta

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓
- b. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.
- c. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.

Pregunta 1

Correcta

Puntuación 1.00

 Marcar  
pregunta

El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Seleccione una:

- a. ... una cota inferior para el valor óptimo que a veces puede ser igual a este. ✓
- b. ... una cota superior para el valor óptimo.
- c. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.

**Pregunta 11**

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

Correcta

Puntúa como 1,00

 Marcar pregunta

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(x^2)$
- b.  $O(1)$  ✓
- c.  $O(x)$

**Pregunta 10**

Correcta

Puntúa como 1,00

 Marcar pregunta

¿ Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores ?

Seleccione una:

- a. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible.
- b. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna. ✓
- c. Ya no se garantizaría la solución óptima pero sí una factible.

Pregunta 12

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Correcta

Puntúa como 1,00

 Marcar pregunta

Seleccione una:

- a. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.
- b. ... tiene un coste temporal asintótico exponencial con respecto al número de objetos.
- c. ... tiene la restricción de que los valores tienen que ser enteros positivos. 

Pregunta 9

En el método voraz ...

Incorrecta

Puntúa como 1,00

 Marcar pregunta

Seleccione una:

- a. ... siempre se encuentra solución pero puede que no sea la óptima. 
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.
- c. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

**Pregunta 8**

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

Correcta

Puntúa como 1,00

 Marcar pregunta

```
unsigned f(unsigned y, unsigned x) { // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A[]
- b. int A
- c. int A[][] ✓

## Pregunta 7

Correcta

Puntúa como 1,00

 Marcar pregunta

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n = 0 \\ \sum_{k=0}^{n-1} T(k) & n > 0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $T(n) \in \Theta(2^n)$  
- b.  $T(n) \in \Theta(n!)$
- c.  $T(n) \in \Theta(n^2)$

**Pregunta 6**

Dado un problema de optimización, el método voraz ...

Correcta

Puntúa como 1,00

 Marcar pregunta

Seleccione una:

- a. ... garantiza la solución óptima sólo para determinados problemas. ✓
- b. ... siempre obtiene la solución óptima.
- c. ... siempre obtiene una solución factible.

**Pregunta 5**

Correcta

Puntúa como 1,00

 Marcar pregunta

¿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?

Seleccione una:

- a. Programación dinámica. 
- b. La estrategia voraz.
- c. Las dos estrategias citadas serían similares en cuanto a eficiencia.

**Pregunta 4**

Correcta

Puntúa como 1,00

 Marcar pregunta

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo divide y vencerás.
- b. un algoritmo de programación dinámica.
- c. un algoritmo voraz. 

**Pregunta 3**

Correcta

Puntúa como 1,00

 Marcar pregunta

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- b. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

Pregunta 2

Correcta

Puntúa como 1,00

 Marcar pregunta

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Hacer una evaluación exhaustiva ``de fuerza bruta'' de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .
- b. Es posible evitar hacer la evaluación exhaustiva ``de fuerza bruta'' guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.
- c. Hacer una evaluación exhaustiva ``de fuerza bruta'' de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ . 

**Pregunta 1**

Correcta

Puntúa como 1,00

 Marcar pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo de programación dinámica. 
- b. ... un algoritmo voraz.
- c. ... un algoritmo del estilo de *divide y vencerás*.

Pregunta 12

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Correcta

Puntúa como 1,00

 Marcar pregunta

Seleccione una:

- a. ... tiene un coste temporal asintótico exponencial con respecto al número de objetos.
- b. ... tiene la restricción de que los valores tienen que ser enteros positivos. ✓
- c. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.

Pregunta 11

Incorrecta

Puntúa como 1,00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$  **X**
- b.  $O(x^2)$
- c.  $O(x)$

Pregunta 10

En el método voraz ...

Correcta

Puntúa como 1,00

 Marcar pregunta

Seleccione una:

- a. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓
- c. ... siempre se encuentra solución pero puede que no sea la óptima.

Pregunta 9

Incorrecta

Puntúa como 1,00

 Marcar  
pregunta

¿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?

Seleccione una:

- a. Programación dinámica.
- b. Las dos estrategias citadas serían similares en cuanto a eficiencia. 
- c. La estrategia voraz.

Pregunta 8

Se pretende implementar mediante programación dinámica recursiva la función recursiva:

Correcta

Puntúa como 1,00

Marcar pregunta

```
float f(unsigned x, int y) {
 if(y < 0) return 0;
 float a = 0.0;
 if (v1[y] <= x)
 a = v2[y] + f(x-v1[y], y-1);
 float b = f(x, y-1);
 return min(a,2+b);
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. unsigned A[] [] ✓
- b. unsigned A
- c. unsigned A[]

Pregunta 7

Correcta

Puntúa como 1,00

Marcar

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

pregunta

- a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.
- b. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓

**Pregunta 6**

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Correcta

Puntúa como 1.00

 Marcar pregunta

Seleccione una:

- a. Ninguna de las otras dos opciones es cierta. ✓
- b. ... la solución óptima está garantizada.
- c. ... al menos una solución factible está garantizada.

**Pregunta 5**

Sin contestar

Puntúa como 1,00

 Marcar pregunta

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- 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 el mínimo de monedas posibles.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.

**Pregunta 4**

Incorrecta

Puntúa como 1,00

 Marcar pregunta

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. La mochila discreta y la asignación de tareas.
- b. La mochila continua y la asignación de tareas.
- c. El fontanero diligente y el problema del cambio. 

Pregunta 3

Correcta

Puntúa como 1,00

 Marcar pregunta

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

- a. El problema de la mochila discreta o sin fraccionamiento. ✓
- b. El problema de la mochila continua o con fraccionamiento.
- c. El árbol de cobertura de coste mínimo de un grafo conexo.

**Pregunta 2**

Correcta

Puntúa como 1,00

 Marcar  
pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo voraz.
- b. ... un algoritmo del estilo de *divide y vencerás*.
- c. ... un algoritmo de programación dinámica. 

Pregunta 1

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Correcta

Puntúa como 1,00

 Marcar pregunta

Seleccione una:

- a. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓
- b. Meter primero los elementos de menor peso.
- c. Meter primero los elementos de mayor valor.

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:



a. El problema de la mochila discreta o sin fraccionamiento.



b. El problema de la mochila continua o con fraccionamiento.



c. El árbol de cobertura de coste mínimo de un grafo conexo.

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.
- b. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.
- c. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y el problema del cambio.
  - b. La mochila continua y la asignación de tareas.
  - c. La mochila discreta y la asignación de tareas.
-

Dado un problema de optimización, el método voraz ...

Seleccione una:

- a. Ninguna de las otras dos opciones es cierta.
- b. ... siempre obtiene la solución óptima.
- c. ... siempre obtiene una solución factible.

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Seleccione una:

- a. ... las decisiones tomadas nunca se reconsideran.
- b. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.
- c. ... con antelación, las posibles decisiones se ordenan de mejor a peor.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A
- b. int A[][]
- c. int A[ ]

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y) {
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A,2+B);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

a.  $O(y^2)$

b.  $O(y)$

c.  $O(1)$

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.
- b. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar.

¿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?

Seleccione una:



a. Programación dinámica.



b. Las dos estrategias citadas serían similares en cuanto a eficiencia.



c. La estrategia voraz.

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Seleccione una:



a. Meter primero los elementos de mayor valor.



b. Meter primero los elementos de menor peso.



c. Meter primero los elementos de mayor valor específico o valor por unidad de peso.

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Seleccione una:

- a. El método voraz.
- b. Programación dinámica.
- c. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Seleccione una:

- 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 un coste temporal asintótico exponencial con respecto al número de objetos.

Pregunta 12

Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Correcta

Puntúa como 1,00

► Marcar

pregunta

Seleccione una:

- a. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.
- b. Programación dinámica. ✓
- c. El método voraz.

Pregunta 11

Correcta

Puntúa como 1.00

 Marcar preguntas

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y) {
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A,2+B);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

a.  $O(y)$  ✓

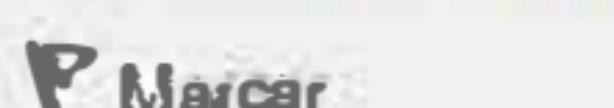
b.  $O(1)$

c.  $O(y^2)$

**Pregunta 10**

Correcta

Puntúa como 1,00



Preguntas

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓
- b. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.
- c. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.

**Pregunta 9**

Correcta

Puntúa como 1.00

 Marcar pregunta

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y la mochila continua. ✓
- b. El fontanero diligente y la asignación de tareas.
- c. El fontanero diligente y el problema del cambio.

Pregunta 8

Correcta

Puntaje como 1,00

 Marcar  
pregunta

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿Qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo divide y vencerás.
- b. un algoritmo voraz. ✓
- c. un algoritmo de programación dinámica.

Pregunta 7

Corrección

Puntúa como 1.00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una

a. int A

b. int A[] ✓

c. int A[][]

**Pregunta 6**

Correcto

Puntuación 1.00

Marked

Pregunta

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

b. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓

c. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.

**Pregunta 5**

Correcta

Puntúa como 1,00

 Marcar pregunta

## La programación dinámica...

Seleccione una:

- a. ... en algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos.
- b. Las otras dos opciones son ciertas. ✓
- c. ... normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores.

Pregunta 4

Sin comienzo

Puntaje como 1,00

 Marcar pregunta

Cuando se calculan los coeficientes binomiales usando la recursión  $\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r}$ , con  $\binom{0}{0} = \binom{n}{n} = 1$ , qué problema se da y cómo se puede resolver?

Seleccione una:

- a. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.
- b. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica.
- c. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.

Pregunta 3

Correcta

Puntúa como 1.00

 Marcar pregunta

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n = 0 \\ \sum_{k=0}^{n-1} T(k) & n > 0 \end{cases}$$

¿cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $T(n) \in \Theta(n^2)$
- b.  $T(n) \in \Theta(n!)$
- c.  $T(n) \in \Theta(2^n)$  ✓

Pregunta 2

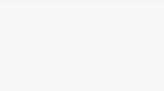
Incorrecta

Puntúa como 1,00

 Marcar pregunta

En el método voraz ...

Seleccione una:

- a. ... siempre se encuentra solución pero puede que no sea la óptima. 
- b. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.
- c. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.

Pregunta 1

Incorrecto

Puntúa como 1.00

 Marcar pregunta

Dado un problema de optimización, el método voraz ...

Seleccione una:

- a. ... siempre obtiene una solución factible. 
- b. Ninguna de las otras dos opciones es cierta.
- c. ... siempre obtiene la solución óptima.

Pregunta 10

Si contestas:

Puntuación como 1.0 0

 Marcar pregunta

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Selezionaone una:

- a. La mochila discreta y la asignación de tareas.
- b. La mochila continua y la asignación de tareas.
- c. El lontanero diligente y el problema del cambio.

Pregunta 9  
Correctas

Puntuación 1.00

Marcar  
Pregunta

¿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?

Seleccione una:

- a. La estrategia voraz.
- b. Las dos estrategias citadas serían similares en cuanto a eficiencia.
- c. Programación dinámica. ✓

Problema 8

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

Sin condensar

```
float f (unsigned x, int y) {
 if (y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A, 2+B);
}
```

Printúa como 1.0 0

Preguntas:  
págimta

¿Cuál es la mayor complejidad espacial que se puede conseguir?

Seleccióne una:

- a.  $O(y)$
- b.  $O(y^2)$
- c.  $O(1)$

Pregunta 7

Correcta

Puntuación 1. 00

■ Marcos  
Preguntas

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Selección una:

- a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ . donde  $n$  es el número de objetos a considerar. ✓
- b. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ . donde  $n$  es el número de objetos a considerar.

Pregunta 6

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

Correcto

unsigned l( unsigned x, unsigned v() ) {

Puntuación 1. 00

```
 if (x == 0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + l(x-k, v));
 return m;
```

Marcar  
pregunta

¿Cuál es la mejor estructura para el almacen?

Selección una:

- a. int x[]()
- b. int x
- c. int x[] ✓

Pregunta 5

Correcto

Puntuación 1.00

P. Marcar  
pregunta

Dada la suma de la recurrencia

$$T(n) = \begin{cases} 1 & n=0 \\ \sum_{k=0}^{n-1} T(k) & n>0 \end{cases}$$

¿Cuál de las siguientes afirmaciones es cierta?

Seleccione una:

- a.  $T(n) \in \Theta(n^2)$
- b.  $T(n) \in \Theta(n!)$
- c.  $T(n) \in \Theta(2^n)$  ✓

Pregunta 4

Correcta

Puntuación 1,00

Mostrar

Pregunta

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarse en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿Qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo de programación dinámica.
- b. un algoritmo divide y vencerás.
- c. un algoritmo voraz. ✓

### Pregunta 3

Correcto

Puntuación 1.00

 Marcar  
preguntas

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo del estilo de *divide y vencerás*.
- b. ... un algoritmo voraz.
- c. ... un algoritmo de programación dinámica. ✓

## Preguntas 2

### En el método voraz ...

Incorrecta

Puntuación 0,00

 Marcar pregunta

Selección una:

- a. ... siempre se encuentra solución pero puede que no sea la óptima. 
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.
- c. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

Pregunta 1

Dado un problema de optimización, el método voraz ...

Correcto

Puntuación 1.00

Marcar  
pregunta

Selección una:

- a. Ninguna de las otras dos opciones es cierta. ✓
- b. ... siempre obtiene una solución factible.
- c. ... siempre obtiene la solución óptima.

## Pregunta 11

Sin contestar

Puntuación 1.00

 Marcar  
pregunta

### La programación dinámica...

Seleccione una:

- a. ... en algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos.
- b. Las otras dos opciones son ciertas.
- c. ... normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores.

Pregunta 12

Incorrecto

Puntaje como 1,00

Marcas  
pregunta

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. La mochila continua y la asignación de tareas. X
- b. El fontanero diligente y el problema del cambio.
- c. La mochila discreta y la asignación de tareas.

Pregunta 11

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Correcto

Puntúa como 1.00

Marcas  
preguntas

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.
- b. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓

Pregunta 10

Correcto

Puntúa como 1,00

 Marcar pregunta

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Seleccione una:

- a. ... al menos una solución factible está garantizada.
- b. Ninguna de las otras dos opciones es cierta. ✓
- c. ... la solución óptima está garantizada.

**Pregunta 9**

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

Correcto

Puntúa como 1.00

 Marcar pregunta

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(x)$
- b.  $O(x^2)$
- c.  $O(1)$  ✓

Pregunta 8  
Correcta

Puntaje como 1,00

 Marcar  
pregunta

¿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?

Seleccione una:

- a. La estrategia voraz.
- b. Las dos estrategias citadas serían similares en cuanto a eficiencia.
- c. Programación dinámica. 

Pregunta 7

Correcta

Puntúa como 1.00

 Marcar pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo voraz.
- b. ... un algoritmo de programación dinámica. ✓
- c. ... un algoritmo del estilo de divide y vencerás.

Pregunta 6

Correcta

Puntúa como 1,00

 Marcar  
preguntas

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

- a. int A[]
- b. int A[][] ✓
- c. int A

Pregunta 5

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Correcto

Puntúa como 1,00

 Marcar pregunta

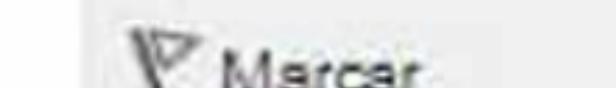
Seleccione una:

- a. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓
- b. Meter primero los elementos de mayor valor.
- c. Meter primero los elementos de menor peso.

**Pregunta 4**

Correcto

Puntúa como 1,00



Marcar pregunta

La mejora que en general aporta la programación dinámica frente a la solución ingenua se consigue gracias al hecho de que ...

Seleccione una:

- 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. ✓

**Pregunta 3**  
Correcta

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Puntúa como 1.00

 Marcar pregunta

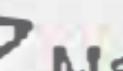
Seleccione una:

- a. Es posible evitar hacer la evaluación exhaustiva "de fuerza bruta" guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.
- b. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ . ✓
- c. Hacer una evaluación exhaustiva "de fuerza bruta" de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .

**Pregunta 2**

Correctas

Puntaje como 1.00

 Marcar pregunta

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

- a. El problema de la mochila continua o con fraccionamiento.
- b. El problema de la mochila discreta o sin fraccionamiento. ✓
- c. El árbol de cobertura de coste mínimo de un grafo conexo.

Pregunta 1

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Correcto

Puntúa como 1.00

 Marcar pregunta

Seleccione una:

- a. .... con antelación, las posibles decisiones se ordenan de mejor a peor.
- b. ... las decisiones tomadas nunca se reconsideran. ✓
- c. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.

Se pretende implementar mediante programacion dinamica iterativa la funcion int f(int x, int y) if( $x \leq y$ ) return 1;  
return  $x + f(x-1,y)$ ; Cual es la mejor complejidad espacial que se puede conseguir

{

$\sim O(x^2)$

$\sim O(x)$

$= O(1)$

}

¿Como se veria afectada la solucion voraz al problema de la asignacion de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores

{

= la solucion factible ya no estaria garantizada, es decir, pudiera ser que el algoritmo no llegue a solucion alguna

~Habria que replantearse el criterio de seleccion para comenzar por aquellos trabajadores con mas restricciones en cuanto a tareas que no pueden realizar

~Ya no se garantizaria la solucion optima pero si una factible

}

En la solucion al problema de la mochila continua ¿por que es conveniente la ordenacion previa de los objetos?

{

= Para reducir la complejidad temporal en la toma de cada decision: de  $O(n)$  a  $O(1)$

~ Para reducir la complejidad temporal en la toma de cada decision: de  $O(n^2)$  a  $O(n \log n)$

~ Porque si no se hace no es posible garantizar la toma de decisiones siga un criterio voraz.

}

Supongamos una solucion recursiva que muestra dos caracteristicas: se basa en obtener soluciones optimas a problemas parciales mas pequeños y por otro, estos subproblemas se resuelve mas de una vez durante el proceso recursivo

{

- ~ un algoritmo voraz

- = un algoritmo de programacion dinamica

- ~ un algoritmo del estilo de divide y vencerás

}

La mejora que en general aporta la programacion dinamica frente a la solucion ingenua se consigue gracias al hecho de que...

{

= ..en la solucion ingenua se resuelve muchas veces un numero relativamente pequeño de subproblemas distintos.

~ ..en la solucion ingenua se resuelve pocas veces un numero relativamente grande de subproblemas distintos.

~ El numero de veces que se resuelven los subproblemas no tiene nada que ver con la eficiencia de los problemas resueltos mediante programacion dinamica

}

De los problemas siguientes... indicad cual no se puede tratar eficientemente como los otros dos

{

~ el problema de cortar un tubo

= el problema de la mochila discreta

~ el problema del cambio

}

Un informatico quiere subir una montaña....

{

~ un algoritmo divide y vencerás.

= un algoritmo voraz.

~ un algoritmo de programación dinámica

}

mediante programacion dinamica recursiva unsigned f(unsigned x, unsigned v[] ).... ¿Cual es la mejor estructura

file:///C/Program%20Files/2o/C%20-%20ADA/OTROS/EXAMENES/PARCIAL%202/ADA%20parcial%202.txt[21/04/2021 18:43:12]

para el almacen?

```
{
~ int a[][]
~ int a
= int a[]
}
```

Dado un problema de optimizacion, el metodo voraz...

{

- ~ ..siempre obtiene la solucion optima
- = Ninguna de las otras dos opciones es cierta

~ ..siempre obtiene una solucion factible

}

Dada la suma de recurrencia....  $T(n)$  sumatorio

{

$\sim \text{pro}(n^2)$

$= \text{pro}(2^n)$

$\sim \text{pro}(n!)$

}

## Coeficientes binomiales ( $n$ $r$ )....

{

~ la recursion puede ser infinita y por tanto necesario organizarla segun el esquema iterativo de programacion

dinamica

~ se repiten muchos calculos y ello se puede evitar haciendo uso de una estrategia voraz

= se repiten muchos calculos y ello se puede evitar usando programacion dinamica

}

En el metodo voraz

{

~ ..el dominio de las decisiones solo pueden ser conjuntos discretos o discretizables

= ..es habitual preparar los datos para disminuir el coste temporal de la funcion que determina cual es la siguiente decision a tomar

~ ..siempre se encuentra la solucion pero puede que no sea la optima

}

El valor que se obtiene con el metodo voraz para el problema de la mochila discreta es..

{

~ ..una cota inferior para el valor optimo, pero que nunca coincide con este

= ..una cota inferior para el valor optimo que a veces puede ser igual a este

~ ..una cota superior para el valor optimo.

}

Dado un problema de optimizacion, el metodo voraz..

{

~ ..siempre obtiene la solucion optima

= ..garantiza la solucion optima solo para determinados problemas

~ ..siempre obtiene una solucion factible

}

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y)
if(y < 0) return 0;
float A = 0.0;
if (v1[y] <= x)
A = v2[y] + f(x-v1[y], y-1);
```

file:///C/Program%20Files/2o/C%20-%20ADA/OTROS/EXAMENES/PARCIAL%202/ADA%20parcial%202.txt[21/04/2021]

```
float B = f(x, y-1);
return min(A,2+B);
```

¿Mejor estructura para el almacen?

```
{
~ unsigned a[]
~ unsigned a
= unsigned a[][]
```

Cuando la descomposicion recursiva de un problema da lugar a subproblemas de tamaño similar ¿que esquema promete ser mas adecuado?

{

= programacion dinamica

~ divide y venceras

~ el metodo voraz

}

Cual de estos tres problemas de optimizacion no tiene, o no se le conoce, solucion voraz optima

{

~ el arbol de cobertura de coste minimo de un grafo conexo

~ mochila continua

= mochila discreta

}

Mediante PD recursiva unsigned f(unsigned x, unsigned v[]) if (x==0) return 0.... ¿Mejor complejidad espacial para el almacen?

{  
~ O(1)  
~ O( $x^2$ )  
= O(x)  
}

Un tubo de  $n$  centímetros....Di cual de estas tres afirmaciones es falsa

{

~ evaluacion exhaustiva de  $\text{pro}(2^n)$ ...

= evaluacion exhaustiva de  $\text{pro}(!n)$ ...

~ evaluacion exhaustiva de  $j < n$ ...

}

# Fibonacci

{

~ Las dos estrategias serian similares en cuanto a eficiencia

= Programacion Dinamica

~ La estrategia Voraz

}

¿Cuál de estas tres estrategias obtiene un mejor valor para la mochila discreta?

{

~ Meter primero los elementos de menor peso

~ Meter primero los elementos de mayor peso

= Meter primero los elementos de mayor valor específico o valor por unidad de peso

}

El problema de encontrar el arbol de recubrimiento de coste minimo para un grafo no dirigido y ponderado...

{

file:///C/Program%20Files/2o/C2%20-%20ADA/OTROS/EXAMENES/PARCIAL%202/ADA%20parcial%202.txt[21/04/2021 18:43:12]

~ ..no se puede resolver en general con una estrategia voraz.

~ solo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vertices del grafo.

= ..se puede resolver siempre con una estrategia voraz.

}

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Seleccione una {

- = ... las decisiones tomadas nunca se reconsideran.
- ~ ... con antelación, las posibles decisiones se ordenan de mejor a peor.
- ~ ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.

}

Si ante un problema de decisión existe un criterio de selección voraz entonces ...

Seleccione una:

{

= Ninguna de las otras dos opciones es cierta.

~ ... la solución óptima está garantizada.

~ ... al menos una solución factible está garantizada.

}

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

{

~Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.

~ $O(n^2)$  a  $O(n \log n)$

=  $O(n)$  a  $O(1)$

}

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x) // suponemos y >= x
if (x==0 || y==x) return 1;
return f(y-1, x-1) + f(y-1, x);
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

{  
~O( $y^2$ )  
=O(y)  
~O(1)  
}

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

{

- = El fontanero diligente y la mochila continua.
- ~ El fontanero diligente y el problema del cambio.
- ~ El fontanero diligente y la asignación de tareas.

}

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[])
```

```
if (x==0)
```

```
return 0;
```

```
unsigned m = 0;
```

```
for (unsigned k = 0; k < x; k++)
```

file:///C/Program%20Files/2o/C2%20-%20ADA/OTROS/EXAMENES/PARCIAL%202/ADA%20parcial%202.txt[21/04]

```
m = max(m, v[k] + f(x-k, v));
```

```
return m;
```

¿

Cuál es la mejor estructura para el almacén?

```
{
```

```
~ int A
```

```
= int A[]
```

```
~ int A[][]
```

```
}
```

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Seleccione una:

{

~ ... tiene un coste temporal asintótico exponencial con respecto al número de objetos.

~ ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.

= ... tiene la restricción de que los valores tienen que ser enteros positivos.

}

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int x, int y)
if(x <= y) return 1;
return x + f(x-1,y);
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

```
{
~ O(x^2)
= O(1)
~ O(x)
}
```

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x) // suponemos y >= x
if (x==0 || y==x) return 1;
return f(y-1, x-1) + f(y-1, x);
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

{

~ O(1)

~ O( $y^2$ )

= O(y)

}

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x) // suponemos y >= x if (x==0 || y==x) return 1;
return f(y-1, x-1) + f(y-1, x);
```

¿Cuál es la mejor estructura para el almacen?

Seleccione una:

```
{
= int a[][]
~ int a
~ int a[]
}
```

La programacion dinamica...

```
{
~ ..normalmente se usa para resolver problemas de optimizacion con dominios discretizables puesto que las tablas se
```

file:///C/Program%20Files/2o/C2%20-%20ADA/OTROS/EXAMENES/PARCIAL%202/ADA%20parcial%202.txt[21/04/2021 18:43:12]

han de indexar con este tipo de valores

~ ..en algunos casos se puede utilizar para resolver problemas de optimizacion con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drasticamente el numero de subproblemas repetidos.  
= Las otras dos opciones son correctas.

```
}
```

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y)
```

```
if(y < 0) return 0;
```

```
float A = 0.0;
```

```
if (v1[y] <= x)
```

```
A = v2[y] + f(x-v1[y], y-1);
```

```
float B = f(x, y-1);
```

```
return min(A,2+B);
```

¿Cuál es la mejor complejidad espacial que se puede conseguir? {

~ O(1)

= O(y)

~ O( $y^2$ )

}

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

{

- = La mochila discreta y la asignación de tareas.
- ~ La mochila continua y la asignación de tareas.
- ~ El fontanero diligente y el problema del cambio.

}

Los algoritmos de programacion dinamica hacen uso..

{

~ ..de que la solucion optima se puede construir añadiendo a la solucion el elemento optimo de los elementos restantes, uno a uno.

= ..de que se puede ahorrar calculos guardando resultados anteriores en un almacen.

~ ..de una estrategia trivial consistente en examinar todas las soluciones posibles.

}

En el método voraz ...

Seleccione una:

- a. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.
- c. ... siempre se encuentra solución pero puede que no sea la óptima.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

a.  $O(y)$

b.  $O(x^2)$

c.  $O(y^2)$



En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar.
- b. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.
- c. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.

La solución óptima al problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. ... se construye haciendo crecer varios árboles que al final acaban injertados en un único árbol.
- b. ... se construye haciendo crecer un único árbol.
- c. ... puede construir un único árbol que va creciendo o bien construir un bosque de árboles que al final se injertan en un único árbol

¿Para qué se utiliza el TAD "Union-find" en el algoritmo de Kruskal?

Seleccione una:

- a. Para comprobar si un vértice ya ha sido visitado.
- b. Para comprobar si un arco forma ciclos.
- c. Para comprobar si dos vértices son equivalentes.

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo voraz.
- b. un algoritmo divide y vencerás.
- c. un algoritmo de programación dinámica.

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x > y) return 1;
 return x*(y-1) + f(x,y-2);
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

Seleccione una:

- a. Ninguna de las otras dos opciones es correcta.
- b. Temporal  $O(x \cdot y)$  y espacial  $O(x)$
- c.  $O(y - x)$ , tanto temporal como espacial.



De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.
- b. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓
- c. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.

Cuando se calculan los coeficientes binomiales usando la recursión  $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$ , con  $\binom{n}{0} = \binom{n}{n} = 1$ , qué problema se da y cómo se puede resolver?

Seleccione una:

- a. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica.
- b. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.
- c. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. La mochila continua y la asignación de tareas.
- b. La mochila discreta y la asignación de tareas.
- c. El fontanero diligente y el problema del cambio.



Los algoritmos de programación dinámica hacen uso ...

Seleccione una:

- a. ... de una estrategia trivial consistente en examinar todas las soluciones posibles.
- b. ... de que se puede ahorrar cálculos guardando resultados anteriores en un almacén.
- c. ... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.

¿Cuál es la mejor complejidad temporal que se puede conseguir?

Seleccione una:

a.  $O(x)$

b.  $O(x \cdot y)$

c.  $O(x^2)$



Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned y = 1; y < x; y++)
 m = max(m, v[y] + f(x-y, v));
 return m;
}
```

¿Qué mecanismo se usa para acelerar el algoritmo de Prim?

Seleccione una:

- a. El TAD "Union-find"
- b. Mantener una lista de los arcos ordenados según su peso.
- c. Mantener para cada vértice su "padre" más cercano.



Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 1; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$
- b.  $O(x)$
- c.  $O(x^2)$



El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Seleccione una:

- a. ... una cota superior para el valor óptimo.
- b. ... una cota inferior para el valor óptimo que a veces puede ser igual a este.
- c. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.



La solución óptima al problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. ... debe construirlo arista a arista: vértice a vértice no puede ser.
- b. ... debe construirlo vértice a vértice: arista a arista no puede ser.
- c. ... puede construirlo tanto vértice a vértice como arista a arista.



En el método voraz ...

Seleccione una:

- a. ... siempre se encuentra solución pero puede que no sea la óptima.
- b. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓
- c. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x > y) return 1;
 return x*(y-1) + f(x,y-2);
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

Seleccione una:

- a.  $O(y - x)$ , tanto temporal como espacial.
- b. Temporal  $O(x \cdot y)$  y espacial  $O(x)$
- c. Ninguna de las otras dos opciones es correcta.



## La solución de programación dinámica iterativa del problema de la mochila discreta ...

Seleccione una:

- a. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.
- b. ... tiene la restricción de que los valores de los objetos tienen que ser números discretos o discretizables.
- c. ... tiene la restricción de que los pesos de los objetos tienen que ser valores discretos o discretizables.



¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. La mochila continua y la asignación de tareas.
- b. El fontanero diligente y el problema del cambio.
- c. La mochila discreta y la asignación de tareas.



De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓
- b. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.
- c. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned y = 1; y < x; y++)
 m = max(m, v[y] + f(x-y, v));
 return m;
}
```

¿Cuál es la mejor complejidad temporal que se puede conseguir?

Seleccione una:

- a.  $O(x)$
- b.  $O(x \cdot y)$
- c.  $O(x^2)$



¿Cuál de estas estrategias voraces obtiene siempre un mejor valor para la mochila discreta?

Seleccione una:

- a. Ninguna de las otras dos opciones es cierta.
- b. Meter primero los elementos de mayor valor específico o valor por unidad de peso.
- c. Meter primero los elementos de mayor valor.



Cuando la descomposición recursiva de un problema da lugar a subproblemas de tamaño similar, ¿qué esquema promete ser más apropiado?

Seleccione una:

- a. El método voraz.
- b. Programación dinámica.
- c. Divide y vencerás, siempre que se garantice que los subproblemas no son del mismo tamaño.

Pregunta 1

Correcta

Puntuación 1.00

sobre 1.00

Marcar

pregunta

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar.
- b. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

Pregunta 2

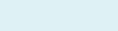
Correcta

Puntuá 1.00  
sobre 1.00

 Desmarcar

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y el problema del cambio.
- b. La mochila continua y la asignación de tareas.
- c. La mochila discreta y la asignación de tareas. 

Pregunta 3

Correcta

Puntuación 1,00  
sobre 1,00

Marcar

Pregunta

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Seleccione una:

- a. ... tiene un coste temporal asintótico exponencial con respecto al número de objetos.
- b. ... tiene la restricción de que los pesos de los objetos tienen que ser valores discretos o discretizables.
- c. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.



Pregunta 4

Correcta

Puntuación 1,00

sobre 1,00

Marcar pregunta

¿Qué mecanismo se usa para acelerar el algoritmo de Prim?

Seleccione una:

- a. El TAD "Union-find"
- b. Mantener una lista de los arcos ordenados según su peso.
- c. Mantener para cada vértice su "padre" más cercano.

Pregunta 5

Correcta

Puntúa 1.00

sobre 1.00

Marcar pregunta

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo del estilo de *divide y vencerás*.
- b. ... un algoritmo de programación dinámica.
- c. ... un algoritmo voraz.



Pregunta 6

Correcta

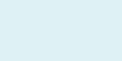
Puntúa 1,00

sobre 1,00

 Marcar  
pregunta

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos.

Seleccione una:

- a. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.
- b. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. 

Preguntas 7

Correcta

Puntuá 1,00  
sobre 1,00

FM Marcar  
pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 1; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

a.  $O(x)$

b.  $O(1)$

c.  $O(x^2)$



Pregunta 8

Correcta

Puntúa 1,00  
sobre 1,00

Marcar  
pregunta

En el método voraz ...

Seleccione una:

- a. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.
- b. ... siempre se encuentra solución pero puede que no sea la óptima.
- c. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.



Pregunta 9

Incorrecta

Puntuación -0,50  
sobre 1,00

Mascara pregunta

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

Seleccione una:

- a.  $O(x - y)$ , tanto temporal como espacial.
- b. Temporal  $O(x - y)$  y espacial  $O(1)$
- c. Ninguna de las otras dos opciones es correcta.



**Pregunta 10**

Correcta

Puntúa 1,00

sobre 1,00

Marcar

pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned y = 1; y < x; y++)
 m = max(m, v[y] + f(x-y, v));
 return m;
}
```

¿Cuál es la mejor complejidad temporal que se puede conseguir?

Seleccione una:

 a.  $O(x)$  b.  $O(x^2)$  c.  $O(x \cdot y)$ 

Pregunta 11

Incorrecta

Puntúa -0,50  
sobre 1,00

Marcar  
pregunta

¿Cuál de estas estrategias voraces obtiene siempre un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de mayor valor.
- b. Meter primero los elementos de mayor valor específico o valor por unidad de peso.
- c. Ninguna de las otras dos opciones es cierta.



Pregunta 12

Sin contestar

Puntúa como  
1,00

 Desmarcar

La solución óptima al problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. ... puede construir un único árbol que va creciendo o bien construir un bosque de árboles que al final se injertan en un único árbol
- b. ... se construye haciendo crecer varios árboles que al final acaban injertados en un único árbol.
- c. ... se construye haciendo crecer un único árbol.

Pregunta 5

Sin responder  
aún

Puntúa como  
1,00

Marcar  
pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad temporal que se puede conseguir?

Seleccione una:

a.  $O(x)$

b.  $O(y)$

c.  $O(x \cdot y)$

d. No contesto (equivalente a no marcar nada).

[Quitar mi elección](#)

Pregunta 4

Sin responder aún

Puntúa como 1,00

Marcar pregunta

¿Qué mecanismo se usa para acelerar el algoritmo de Prim?

Seleccione una:

- a. Mantener para cada vértice el vértice origen de la arista más corta hasta él.
- b. No contesto (equivalente a no marcar nada).
- c. Mantener una lista de los arcos ordenados según su peso.
- d. El TAD "Union-find"

Ficha 12

Sin responder  
aún

Puntúa como  
1,00

■ Marcar  
pregunta

Los algoritmos de programación dinámica hacen uso ...

Seleccione una:

- a. ... de una estrategia que ahorra cálculos innecesarios.
- b. ... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.
- c. ... de una estrategia trivial consistente en examinar todas las soluciones posibles.
- d. No contesto (equivalente a no marcar nada).

[Quitar mi elección](#)

Pregunta 8

Sin responder aún

Puntúa como 1,00

¶ Marcar pregunta

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x > y) return 1;
 return x + f(x,y-2);
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

Seleccione una:

- a.  $O(y - x)$ , tanto temporal como espacial.
- b. Ninguna de las otras dos opciones es correcta.
- c. Temporal  $O(x \cdot y)$  y espacial  $O(x)$
- d. No contesto (equivalente a no marcar nada).



Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A,2+B);
}
```

¿Cuál es la mejor complejidad temporal que se puede conseguir?

Seleccione una:

- a.  $O(y)$
- b.  $O(x \cdot y)$
- c. No contesto (equivalente a no marcar nada).
- d.  $O(x)$

Pregunta 1

Correcta

Puntúa 1,00  
sobre 1,00

¿Para qué se utiliza el TAD "Union-find" en el algoritmo de Kruskal?

Seleccione una:

- a. Para comprobar si un vértice ya ha sido visitado.
- b. Para comprobar si un arco forma ciclos. ✓
- c. Para comprobar si dos vértices son equivalentes.
- d. No contesto (equivalente a no marcar nada).

Pregunta **2**

Correcta

Puntúa 1,00  
sobre 1,00

El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Seleccione una:

- a. ... una cota superior para el valor óptimo.
- b. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
- c. ... una cota inferior para el valor óptimo que a veces puede ser igual a este.
- d. No contesto (equivalente a no marcar nada).



Pregunta 3

Correcta

Puntúa 1,00  
sobre 1,00

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Hacer una evaluación exhaustiva ``de fuerza bruta'' de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ . ✓
- b. No contesto (equivalente a no marcar nada).
- c. Es posible evitar hacer la evaluación exhaustiva ``de fuerza bruta'' guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.
- d. Hacer una evaluación exhaustiva ``de fuerza bruta'' de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .

Pregunta **4**

Correcta

Puntúa 1,00  
sobre 1,00

La solución óptima al problema de encontrar el **árbol de recubrimiento de coste mínimo** para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. ... se construye haciendo crecer un único **árbol**.
- b. ... puede construir un único **árbol** que va creciendo o bien construir un bosque de árboles que al final se injertan en un único **árbol** ✓
- c. ... se construye haciendo crecer varios árboles que al final acaban injertados en un único **árbol**.
- d. No contesto (equivalente a no marcar nada).

Pregunta **5**

Correcta

Puntúa 1,00  
sobre 1,00

¿Cuál de estas estrategias voraces obtiene siempre un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de mayor valor específico o valor por unidad de peso.
- b. No contesto (equivalente a no marcar nada).
- c. Meter primero los elementos de mayor valor.
- d. Ninguna de las otras dos opciones es cierta.



Pregunta **6**

Correcta

Puntúa 1,00  
sobre 1,00

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.
- c. ... las decisiones tomadas nunca se reconsideran.
- d. ... con antelación, las posibles decisiones se ordenan de mejor a peor.



Pregunta **7**

Correcta

Puntúa 1,00  
sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(1)$
- b. No contesto (equivalente a no marcar nada).
- c.  $O(y)$
- d.  $O(y^2)$



Pregunta 8

Correcta

Puntúa 1,00  
sobre 1,00

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x > y) return 1;
 return x + f(x,y-2);
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

Seleccione una:

- a. Temporal  $O(x \cdot y)$  y espacial  $O(x)$
- b. No contesto (equivalente a no marcar nada).
- c.  $O(y - x)$ , tanto temporal como espacial.
- d. Ninguna de las otras dos opciones es correcta.



Pregunta **9**

Correcta

Puntúa 1,00  
sobre 1,00

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... tiene la restricción de que los pesos de los objetos tienen que ser números discretos o discretizables. ✓
- c. ... tiene la restricción de que los valores de los objetos tienen que ser números discretos o discretizables.
- d. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.

Pregunta 10

Correcta

Puntúa 1,00  
sobre 1,00

El problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo.
- b. No contesto (equivalente a no marcar nada).
- c. ... se puede resolver siempre con una estrategia voraz.
- d. ... no se puede resolver en general con una estrategia voraz.



Pregunta 11

Correcta

Puntúa 1,00  
sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned y = 0; y < x; y++)
 m = max(m, v[y] + f(x-y, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(x \cdot y)$
- b.  $O(x)$
- c. No contesto (equivalente a no marcar nada).
- d.  $O(y)$



Pregunta **12**

Correcta

Puntúa 1,00  
sobre 1,00

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo voraz.
- b. No contesto (equivalente a no marcar nada).
- c. ... un algoritmo del estilo de *divide y vencerás*.
- d. ... un algoritmo de programación dinámica.



Pregunta 1

Correcta

Puntúa 1,00 sobre 1,00

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. El problema de la mochila discreta o sin fraccionamiento.
- c. El problema de la mochila continua o con fraccionamiento.
- d. El árbol de cobertura de coste mínimo de un grafo conexo.



Pregunta **2**

Correcta

Puntúa 1,00 sobre 1,00

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Seleccione una:

- a. ... con antelación, las posibles decisiones se ordenan de mejor a peor.
- b. No contesto (equivalente a no marcar nada).
- c. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.
- d. ... las decisiones tomadas nunca se reconsideran.



Pregunta 3

Correcta

Puntúa 1,00 sobre 1,00

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x > y) return 1;
 return x*(y-1) + f(x,y-2);
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

Seleccione una:

- a.  $O(y - x)$ , tanto temporal como espacial. ✓
- b. Temporal  $O(x \cdot y)$  y espacial  $O(x)$
- c. No contesto (equivalente a no marcar nada).
- d. Ninguna de las otras dos opciones es correcta.

Pregunta 4

Correcta

Puntúa 1,00 sobre 1,00

La solución óptima al problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... puede construir un único árbol que va creciendo o bien construir un bosque de árboles que al final se injertan en un único árbol ✓
- c. ... se construye haciendo crecer varios árboles que al final acaban injertados en un único árbol.
- d. ... se construye haciendo crecer un único árbol.

Pregunta **5**

Correcta

Puntúa 1,00 sobre 1,00

Los algoritmos de programación dinámica hacen uso ...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... de una estrategia trivial consistente en examinar todas las soluciones posibles.
- c. ... de una estrategia que ahorra cálculos innecesarios. 
- d. ... de que la solución óptima se puede construir añadiendo a la solución el elemento óptimo de los elementos restantes, uno a uno.

Pregunta **6**

Correcta

Puntúa 1,00 sobre 1,00

¿Qué mecanismo se usa para acelerar el algoritmo de Prim?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. El TAD "Union-find"
- c. Mantener para cada vértice el vértice origen de la arista más corta hasta él.
- d. Mantener una lista de los arcos ordenados según su peso.



Pregunta 7

Correcta

Puntúa 1,00 sobre 1,00

Un tubo de  $n$  centímetros de largo se puede cortar en segmentos de 1 centímetro, 2 centímetros, etc. Existe una lista de los precios a los que se venden los segmentos de cada longitud. Una de las maneras de cortar el tubo es la que más ingresos nos producirá. Di cuál de estas tres afirmaciones es falsa.

Seleccione una:

- a. Hacer una evaluación exhaustiva ``de fuerza bruta'' de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(n!)$ . ✓
- b. Hacer una evaluación exhaustiva ``de fuerza bruta'' de todas las posibles maneras de cortar el tubo consume un tiempo  $\Theta(2^n)$ .
- c. Es posible evitar hacer la evaluación exhaustiva ``de fuerza bruta'' guardando, para cada posible longitud  $j < n$  el precio más elevado posible que se puede obtener dividiendo el tubo correspondiente.
- d. No contesto (equivalente a no marcar nada).

Pregunta **8**

Correcta

Puntúa 1,00 sobre 1,00

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo de programación dinámica. ✓
- b. ... un algoritmo del estilo de *divide y vencerás*.
- c. No contesto (equivalente a no marcar nada).
- d. ... un algoritmo voraz.

Pregunta **10**

Correcta

Puntúa 1,00 sobre 1,00

El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Seleccione una:

- a. ... una cota superior para el valor óptimo.
- b. ... una cota inferior para el valor óptimo que a veces puede ser igual a este.
- c. No contesto (equivalente a no marcar nada).
- d. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.



Pregunta 11

Correcta

Puntúa 1,00 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad temporal que se puede conseguir?

Seleccione una:

- a.  $O(x \cdot y)$
- b.  $O(y)$
- c.  $O(x)$
- d. No contesto (equivalente a no marcar nada).



Pregunta **12**

Correcta

Puntúa 1,00 sobre 1,00

El problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo.
- b. ... no se puede resolver en general con una estrategia voraz.
- c. ... se puede resolver siempre con una estrategia voraz. ✓
- d. No contesto (equivalente a no marcar nada).

Pregunta 1

Correcta

Puntúa 1,00 sobre 1,00

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo del estilo de *divide y vencerás*.
- b. No contesto (equivalente a no marcar nada).
- c. ... un algoritmo de programación dinámica.
- d. ... un algoritmo voraz.

Pregunta 2

Correcta

Puntúa 1,00 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A,2+B);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

a.  $O(y)$



b. No contesto (equivalente a no marcar nada).

c.  $O(1)$

d.  $O(y^2)$

Pregunta 3

Correcta

Puntúa 1,00 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned y = 0; y < x; y++)
 m = max(m, v[y] + f(x-y, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(y)$
- b.  $O(x \cdot y)$
- c. No contesto (equivalente a no marcar nada).
- d.  $O(x)$



Pregunta 4

Correcta

Puntúa 1,00 sobre 1,00

¿Qué mecanismo se usa para acelerar el algoritmo de Prim?

Seleccione una:

- a. Mantener para cada vértice el vértice origen de la arista más corta hasta él.
- b. No contesto (equivalente a no marcar nada).
- c. El TAD "Union-find"
- d. Mantener una lista de los arcos ordenados según su peso.



Pregunta **5**

Correcta

Puntúa 1,00 sobre 1,00

La solución óptima al problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. ... se construye haciendo crecer varios árboles que al final acaban injertados en un único árbol.
- b. ... se construye haciendo crecer un único árbol.
- c. ... puede construir un único árbol que va creciendo o bien construir un bosque de árboles que al final se injertan en un único árbol ✓
- d. No contesto (equivalente a no marcar nada).

Pregunta **6**

Correcta

Puntúa 1,00 sobre 1,00

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- b. No contesto (equivalente a no marcar nada).
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓
- d. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

Pregunta **7**

Correcta

Puntúa 1,00 sobre 1,00

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. La mochila continua y la asignación de tareas.
- b. No contesto (equivalente a no marcar nada).
- c. El fontanero diligente y el problema del cambio.
- d. La mochila discreta y la asignación de tareas.



Pregunta 8

Correcta

Puntúa 1,00 sobre 1,00

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x > y) return 1;
 return x + f(x,y-2);
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

Seleccione una:

- a.  $O(y - x)$ , tanto temporal como espacial.
- b. Ninguna de las otras dos opciones es correcta.
- c. Temporal  $O(x \cdot y)$  y espacial  $O(x)$
- d. No contesto (equivalente a no marcar nada).



Pregunta 9

Correcta

Puntúa 1,00 sobre 1,00

En el método voraz ...

Seleccione una:

- a. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.
- b. No contesto (equivalente a no marcar nada).
- c. ... siempre se encuentra solución pero puede que no sea la óptima.
- d. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.



Pregunta **10**

Correcta

Puntúa 1,00 sobre 1,00

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- 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 el mínimo de monedas posibles.
- c. No contesto (equivalente a no marcar nada).
- d. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓

Pregunta **11**

Correcta

Puntúa 1,00 sobre 1,00

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

- a. El problema de la mochila continua o con fraccionamiento.
- b. No contesto (equivalente a no marcar nada).
- c. El árbol de cobertura de coste mínimo de un grafo conexo.
- d. El problema de la mochila discreta o sin fraccionamiento.



Pregunta **12**

Correcta

Puntúa 1,00 sobre 1,00

La programación dinámica...

Seleccione una:

- a. ... en algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos.
- b. No contesto (equivalente a no marcar nada).
- c. Las otras dos opciones son ciertas. ✓
- d. ... normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores.

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

Seleccione una:

- a. Ninguna de las otras dos opciones es correcta. ✗
- b. No contesto (equivalente a no marcar nada).
- c.  $O(x - y)$ , tanto temporal como espacial.
- d. Temporal  $O(x - y)$  y espacial  $O(1)$

Pregunta **2**

Correcta

Puntúa 1,00 sobre 1,00

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... las decisiones tomadas nunca se reconsideran.
- c. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.
- d. ... con antelación, las posibles decisiones se ordenan de mejor a peor.



Pregunta 3

Incorrecta

Puntúa -0,50 sobre 1,00

Cuando se calculan los coeficientes binomiales usando la recursión  $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$ , con  $\binom{n}{0} = \binom{n}{n} = 1$ , qué problema se da y cómo se puede resolver?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica. ✗
- c. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.
- d. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica.

Pregunta **4**

Correcta

Puntúa 1,00 sobre 1,00

¿Para qué se utiliza el TAD "Union-find" en el algoritmo de Kruskal?

Seleccione una:

a. Para comprobar si un arco forma ciclos.



b. Para comprobar si un vértice ya ha sido visitado.

c. No contesto (equivalente a no marcar nada).

d. Para comprobar si dos vértices son equivalentes.

Pregunta 5

Correcta

Puntúa 1,00 sobre 1,00

El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Seleccione una:

- a. ... una cota inferior para el valor óptimo que a veces puede ser igual a este.
- b. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
- c. ... una cota superior para el valor óptimo.
- d. No contesto (equivalente a no marcar nada).



Pregunta **6**

Incorrecta

Puntúa -0,50 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad temporal que se puede conseguir?

Seleccione una:

- a.  $O(x)$
- b. No contesto (equivalente a no marcar nada).
- c.  $O(x \cdot y)$
- d.  $O(y)$

✗

Pregunta **7**

Correcta

Puntúa 1,00 sobre 1,00

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y el problema del cambio.
- b. El fontanero diligente y la asignación de tareas.
- c. El fontanero diligente y la mochila continua.
- d. No contesto (equivalente a no marcar nada).



Pregunta **8**

Correcta

Puntúa 1,00 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(y^2)$
- b. No contesto (equivalente a no marcar nada).
- c.  $O(y)$
- d.  $O(1)$



Pregunta **9**

Correcta

Puntúa 1,00 sobre 1,00

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Seleccione una:

- a. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.
- b. ... tiene la restricción de que los valores de los objetos tienen que ser números discretos o discretizables.
- c. No contesto (equivalente a no marcar nada).
- d. ... tiene la restricción de que los pesos de los objetos tienen que ser números discretos o discretizables.



Pregunta **10**

Correcta

Puntúa 1,00 sobre 1,00

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓
- b. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.
- c. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.
- d. No contesto (equivalente a no marcar nada).

Pregunta **11**

Correcta

Puntúa 1,00 sobre 1,00

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo divide y vencerás.
- b. un algoritmo de programación dinámica.
- c. un algoritmo voraz.
- d. No contesto (equivalente a no marcar nada).



Pregunta **12**

Correcta

Puntúa 1,00 sobre 1,00

La solución óptima al problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. ... puede construirlo tanto vértice a vértice como arista a arista.
- b. ... debe construirlo arista a arista: vértice a vértice no puede ser.
- c. ... debe construirlo vértice a vértice: arista a arista no puede ser.
- d. No contesto (equivalente a no marcar nada).



Pregunta **1**

Correcta

Puntúa 1,00 sobre 1,00

La programación dinámica...

Seleccione una:

- a. ... en algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos.
- b. ... normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores.
- c. No contesto (equivalente a no marcar nada).
- d. Las otras dos opciones son ciertas. ✓

Pregunta 2

Correcta

Puntúa 1,00 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned y = 0; y < x; y++)
 m = max(m, v[y] + f(x-y, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $O(x)$
- c.  $O(y)$
- d.  $O(x \cdot y)$



Pregunta 3

Correcta

Puntúa 1,00 sobre 1,00

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.
- b. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓
- c. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.
- d. No contesto (equivalente a no marcar nada).

Pregunta 4

Correcta

Puntúa 1,00 sobre 1,00

El problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. ... no se puede resolver en general con una estrategia voraz.
- b. No contesto (equivalente a no marcar nada).
- c. sólo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vértices del grafo.
- d. ... se puede resolver siempre con una estrategia voraz.



Pregunta **5**

Correcta

Puntúa 1,00 sobre 1,00

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

- a. El problema de la mochila continua o con fraccionamiento.
- b. El problema de la mochila discreta o sin fraccionamiento.
- c. No contesto (equivalente a no marcar nada).
- d. El árbol de cobertura de coste mínimo de un grafo conexo.



Pregunta **6**

Correcta

Puntúa 1,00 sobre 1,00

¿Para qué se utiliza el TAD "Union-find" en el algoritmo de Kruskal?

Seleccione una:

- a. Para comprobar si un vértice ya ha sido visitado.
- b. No contesto (equivalente a no marcar nada).
- c. Para comprobar si un arco forma ciclos.
- d. Para comprobar si dos vértices son equivalentes.



Pregunta 7

Correcta

Puntúa 1,00 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

a.  $O(1)$



b. No contesto (equivalente a no marcar nada).

c.  $O(x^2)$

d.  $O(x)$

Pregunta 8

Correcta

Puntúa 1,00 sobre 1,00

¿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?

Seleccione una:

- a. La estrategia voraz.
- b. Las dos estrategias citadas serían similares en cuanto a eficiencia.
- c. Programación dinámica. ✓
- d. No contesto (equivalente a no marcar nada).

Pregunta **9**

Correcta

Puntúa 1,00 sobre 1,00

La solución óptima al problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. ... debe construirlo vértice a vértice: arista a arista no puede ser.
- b. ... puede construirlo tanto vértice a vértice como arista a arista. ✓
- c. ... debe construirlo arista a arista: vértice a vértice no puede ser.
- d. No contesto (equivalente a no marcar nada).

Pregunta **10**

Correcta

Puntúa 1,00 sobre 1,00

El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Seleccione una:

- a. ... una cota inferior para el valor óptimo que a veces puede ser igual a este.
- b. ... una cota superior para el valor óptimo.
- c. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
- d. No contesto (equivalente a no marcar nada).



Pregunta **11**

Correcta

Puntúa 1,00 sobre 1,00

En el método voraz ...

Seleccione una:

- a. ... siempre se encuentra solución pero puede que no sea la óptima.
- b. No contesto (equivalente a no marcar nada).
- c. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓
- d. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.

Pregunta **12**

Correcta

Puntúa 1,00 sobre 1,00

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x > y) return 1;
 return x + f(x,y-2);
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

Seleccione una:

- a.  $O(y - x)$ , tanto temporal como espacial.
- b. Ninguna de las otras dos opciones es correcta.
- c. Temporal  $O(x \cdot y)$  y espacial  $O(x)$
- d. No contesto (equivalente a no marcar nada).



Pregunta 1

Correcta

Puntúa 1,00 sobre 1,00

¿ Cómo se vería afectada la solución voraz al problema de la asignación de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores ?

Seleccione una:

- a. Ya no se garantizaría la solución óptima pero sí una factible.
- b. La solución factible ya no estaría garantizada, es decir, pudiera ser que el algoritmo no llegue a solución alguna. 
- c. Habría que replantearse el criterio de selección para comenzar por aquellos trabajadores con más restricciones en cuanto a las tareas que no pueden realizar para asegurar, al menos, una solución factible.
- d. No contesto (equivalente a no marcar nada).

Pregunta **2**

Correcta

Puntúa 1,00 sobre 1,00

En el método voraz ...

Seleccione una:

- a. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar. ✓
- b. No contesto (equivalente a no marcar nada).
- c. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.
- d. ... siempre se encuentra solución pero puede que no sea la óptima.

Pregunta 3

Correcta

Puntúa 1,00 sobre 1,00

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x * f(x-1,y) + y;
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

Seleccione una:

a. No contesto (equivalente a no marcar nada).

b. Temporal  $O(x - y)$  y espacial  $O(1)$

c.  $O(x - y)$ , tanto temporal como espacial. 

d. Ninguna de las otras dos opciones es correcta.

Pregunta **4**

Incorrecta

Puntúa -0,50 sobre 1,00

La solución de programación dinámica iterativa del problema de la mochila discreta ...

Seleccione una:

- a. ... tiene la restricción de que los valores de los objetos tienen que ser números discretos o discretizables. ✗
- b. No contesto (equivalente a no marcar nada).
- c. ... tiene la restricción de que los pesos de los objetos tienen que ser números discretos o discretizables.
- d. ... calcula menos veces el valor de la mochila que la correspondiente solución de programación dinámica recursiva.

Pregunta 5

Correcta

Puntúa 1,00 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

a.  $O(x)$



b.  $O(1)$

c. No contesto (equivalente a no marcar nada).

d.  $O(x^2)$

Pregunta **6**

Correcta

Puntúa 1,00 sobre 1,00

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo voraz.
- b. No contesto (equivalente a no marcar nada).
- c. ... un algoritmo de programación dinámica.
- d. ... un algoritmo del estilo de *divide y vencerás*.



Pregunta 7

Correcta

Puntúa 1,00 sobre 1,00

La solución óptima al problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. ... se construye haciendo crecer varios árboles que al final acaban injertados en un único árbol.
- b. ... puede construir un único árbol que va creciendo o bien construir un bosque de árboles que al final se injertan en un único árbol ✓
- c. No contesto (equivalente a no marcar nada).
- d. ... se construye haciendo crecer un único árbol.

Pregunta **8**

Incorrecta

Puntúa -0,50 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int x, int y) {
 if(x <= y) return 1;
 return x + f(x-1,y);
}
```

¿Cuál es la mejor complejidad temporal que se puede conseguir?

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b.  $O(x)$
- c.  $O(y)$
- d.  $O(x \cdot y)$



Pregunta **9**

Correcta

Puntúa 1,00 sobre 1,00

¿Cuál de estas estrategias voraces obtiene siempre un mejor valor para la mochila discreta?

Seleccione una:

- a. Ninguna de las otras dos opciones es cierta. ✓
- b. Meter primero los elementos de mayor valor específico o valor por unidad de peso.
- c. Meter primero los elementos de mayor valor.
- d. No contesto (equivalente a no marcar nada).

Pregunta **10**

Correcta

Puntúa 1,00 sobre 1,00

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓
- d. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.

Pregunta **11**

Correcta

Puntúa 1,00 sobre 1,00

El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Seleccione una:

- a. ... una cota inferior para el valor óptimo que a veces puede ser igual a este.
- b. ... una cota superior para el valor óptimo.
- c. No contesto (equivalente a no marcar nada).
- d. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.



Pregunta **12**

Correcta

Puntúa 1,00 sobre 1,00

¿Para qué se utiliza el TAD "Union-find" en el algoritmo de Kruskal?

Seleccione una:

- a. Para comprobar si un vértice ya ha sido visitado.
- b. No contesto (equivalente a no marcar nada).
- c. Para comprobar si dos vértices son equivalentes.
- d. Para comprobar si un arco forma ciclos.



Pregunta 1

Correcta

Puntúa 1,00 sobre 1,00

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x > y) return 1;
 return x*(y-1) + f(x,y-2);
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

Seleccione una:

- a. Ninguna de las otras dos opciones es correcta.
- b. No contesto (equivalente a no marcar nada).
- c. Temporal  $O(x \cdot y)$  y espacial  $O(x)$
- d.  $O(y - x)$ , tanto temporal como espacial.



Pregunta **2**

Correcta

Puntúa 1,00 sobre 1,00

La solución óptima al problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. ... se construye haciendo crecer varios árboles que al final acaban injertados en un único árbol.
- b. No contesto (equivalente a no marcar nada).
- c. ... puede construir un único árbol que va creciendo o bien construir un bosque de árboles que al final se injertan en un único árbol ✓
- d. ... se construye haciendo crecer un único árbol.

Pregunta **3**

Correcta

Puntúa 1,00 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y){
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A,2+B);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(y^2)$
- b.  $O(y)$
- c. No contesto (equivalente a no marcar nada).
- d.  $O(1)$



Pregunta 4

Correcta

Puntúa 1,00 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned y = 0; y < x; y++)
 m = max(m, v[y] + f(x-y, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(x)$
- b.  $O(x \cdot y)$
- c. No contesto (equivalente a no marcar nada).
- d.  $O(y)$



Pregunta **5**

Correcta

Puntúa 1,00 sobre 1,00

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- a. El problema del cambio, o sea, el de encontrar la manera de entregar una cantidad de dinero usando el mínimo de monedas posibles.
- b. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores. ✓
- c. No contesto (equivalente a no marcar nada).
- d. El problema de cortar un tubo de forma que se obtenga el máximo beneficio posible.

Pregunta **6**

Correcta

Puntúa 1,00 sobre 1,00

¿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?

Seleccione una:

- a. Las dos estrategias citadas serían similares en cuanto a eficiencia.
- b. Programación dinámica.
- c. La estrategia voraz.
- d. No contesto (equivalente a no marcar nada).



Pregunta 7

Correcta

Puntúa 1,00 sobre 1,00

Un informático quiere subir a una montaña y para ello decide que tras cada paso, el siguiente debe tomarlo en la dirección de máxima pendiente hacia arriba. Además, entenderá que ha alcanzado la cima cuando llegue a un punto en el que no haya ninguna dirección que sea cuesta arriba. ¿qué tipo de algoritmo está usando nuestro informático?

Seleccione una:

- a. un algoritmo divide y vencerás.
- b. un algoritmo voraz. ✓
- c. No contesto (equivalente a no marcar nada).
- d. un algoritmo de programación dinámica.

Pregunta 8

Correcta

Puntúa 1,00 sobre 1,00

La programación dinámica...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. Las otras dos opciones son ciertas. ✓
- c. ... en algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos.
- d. ... normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores.

Pregunta **9**

Correcta

Puntúa 1,00 sobre 1,00

¿Para qué se utiliza el TAD "Union-find" en el algoritmo de Kruskal?

Seleccione una:

- a. Para comprobar si un vértice ya ha sido visitado.
- b. No contesto (equivalente a no marcar nada).
- c. Para comprobar si dos vértices son equivalentes.
- d. Para comprobar si un arco forma ciclos.



Pregunta **10**

Correcta

Puntúa 1,00 sobre 1,00

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... con antelación, las posibles decisiones se ordenan de mejor a peor.
- c. ... antes de tomar una decisión se comprueba si satisface las restricciones del problema.
- d. ... las decisiones tomadas nunca se reconsideran.



Pregunta **11**

Correcta

Puntúa 1,00 sobre 1,00

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. El fontanero diligente y el problema del cambio.
- b. La mochila discreta y la asignación de tareas.
- c. No contesto (equivalente a no marcar nada).
- d. La mochila continua y la asignación de tareas.



Pregunta **12**

Correcta

Puntúa 1,00 sobre 1,00

El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...

Seleccione una:

- a. No contesto (equivalente a no marcar nada).
- b. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
- c. ... una cota inferior para el valor óptimo que a veces puede ser igual a este.
- d. ... una cota superior para el valor óptimo.



**Comenzado el** martes, 4 de mayo de 2021, 15:06

**Estado** Finalizado

**Finalizado en** martes, 4 de mayo de 2021, 15:18

**Tiempo empleado** 12 minutos 15 segundos

**Puntos** 12,00/12,00

**Calificación** **10,00** de 10,00 (**100%**)

Pregunta 1

Correcta

Puntúa 1,00 sobre 1,00

Supongamos que una solución recursiva a un problema de optimización muestra estas dos características: por un lado, se basa en obtener soluciones óptimas a problemas parciales más pequeños, y por otro, estos subproblemas se resuelven más de una vez durante el proceso recursivo. Este problema es candidato a tener una solución alternativa basada en ...

Seleccione una:

- a. ... un algoritmo del estilo de *divide y vencerás*.
- b. No contesto (equivalente a no marcar nada).
- c. ... un algoritmo de programación dinámica.
- d. ... un algoritmo voraz.



Pregunta 2

Correcta

Puntúa 1,00 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
float f(unsigned x, int y) {
 if(y < 0) return 0;
 float A = 0.0;
 if (v1[y] <= x)
 A = v2[y] + f(x-v1[y], y-1);
 float B = f(x, y-1);
 return min(A,2+B);
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(y)$
- b. No contesto (equivalente a no marcar nada).
- c.  $O(1)$
- d.  $O(y^2)$



## Pregunta 3

Correcta

Puntúa 1,00 sobre 1,00

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned y = 0; y < x; y++)
 m = max(m, v[y] + f(x-y, v));
 return m;
}
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

- a.  $O(y)$
  - b.  $O(x \cdot y)$
  - c. No contesto (equivalente a no marcar nada).
  - d.  $O(x)$
- ✓

## Pregunta 4

Correcta

Puntúa 1,00 sobre 1,00

¿Qué mecanismo se usa para acelerar el algoritmo de Prim?

Seleccione una:

- a. Mantener para cada vértice el vértice origen de la arista más corta hasta él.
  - b. No contesto (equivalente a no marcar nada).
  - c. El TAD "Union-find"
  - d. Mantener una lista de los arcos ordenados según su peso.
- ✓

## Pregunta 5

Correcta

Puntúa 1,00 sobre 1,00

La solución óptima al problema de encontrar el árbol de recubrimiento de coste mínimo para un grafo no dirigido, conexo y ponderado ...

Seleccione una:

- a. ... se construye haciendo crecer varios árboles que al final acaban injertados en un único árbol.
  - b. ... se construye haciendo crecer un único árbol.
  - c. ... puede construir un único árbol que va creciendo o bien construir un bosque de árboles que al final se injertan en un único árbol
  - d. No contesto (equivalente a no marcar nada).
- ✓

## Pregunta 6

Correcta

Puntúa 1,00 sobre 1,00

En la solución al problema de la mochila continua ¿por qué es conveniente la ordenación previa de los objetos?

Seleccione una:

- a. Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.
- b. No contesto (equivalente a no marcar nada).
- c. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n)$  a  $O(1)$ , donde  $n$  es el número de objetos a considerar. ✓
- d. Para reducir la complejidad temporal en la toma de cada decisión: de  $O(n^2)$  a  $O(n \log n)$ , donde  $n$  es el número de objetos a considerar.

## Pregunta 7

Correcta

Puntúa 1,00 sobre 1,00

¿Cuál de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución (óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

- a. La mochila continua y la asignación de tareas.
- b. No contesto (equivalente a no marcar nada).
- c. El fontanero diligente y el problema del cambio.
- d. La mochila discreta y la asignación de tareas. ✓

## Pregunta 8

Correcta

Puntúa 1,00 sobre 1,00

Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int x, int y) {
 if(x > y) return 1;
 return x + f(x,y-2);
}
```

En el caso más desfavorable, ¿qué complejidades temporal y espacial cabe esperar de la función resultante?

Seleccione una:

- a.  $O(y - x)$ , tanto temporal como espacial. ✓
- b. Ninguna de las otras dos opciones es correcta.
- c. Temporal  $O(x \cdot y)$  y espacial  $O(x)$
- d. No contesto (equivalente a no marcar nada).

**Pregunta 9**

Correcta

Puntúa 1,00 sobre 1,00

En el método voraz ...

Seleccione una:

- a. ... el dominio de las decisiones sólo pueden ser conjuntos discretos o discretizables.
- b. No contesto (equivalente a no marcar nada).
- c. ... siempre se encuentra solución pero puede que no sea la óptima.
- d. ... es habitual preparar los datos para disminuir el coste temporal de la función que determina cuál es la siguiente decisión a tomar.

**Pregunta 10**

Correcta

Puntúa 1,00 sobre 1,00

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- 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 el mínimo de monedas posibles.
- c. No contesto (equivalente a no marcar nada).
- d. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.

**Pregunta 11**

Correcta

Puntúa 1,00 sobre 1,00

¿Cuál de estos tres problemas de optimización no tiene, o no se le conoce, una solución voraz óptima?

Seleccione una:

- a. El problema de la mochila continua o con fraccionamiento.
- b. No contesto (equivalente a no marcar nada).
- c. El árbol de cobertura de coste mínimo de un grafo conexo.
- d. El problema de la mochila discreta o sin fraccionamiento.



**Pregunta 12**

Correcta

Puntúa 1,00 sobre 1,00

La programación dinámica...

Seleccione una:

- a. ... en algunos casos se puede utilizar para resolver problemas de optimización con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente el número de subproblemas repetidos.
- b. No contesto (equivalente a no marcar nada).
- c. Las otras dos opciones son ciertas. ✓
- d. ... normalmente se usa para resolver problemas de optimización con dominios discretizables puesto que las tablas se han de indexar con este tipo de valores.

[◀ Primer parcial](#)

[Ir a...](#)

## Parcial2

Se pretende implementar mediante programacion dinamica iterativa la funcion

```
int f(int x, int y) if(x<=y) return 1; return x + f(x-1,y);
```

Cual es la mejor complejidad espacial que se puede conseguir

```
{
~O(x^2)
~O(x)
=O(1)
}
```

Como se veria afectada la solucion voraz al problema de la asignacion de tareas en el caso de que se incorporaran restricciones que contemplen que ciertas tareas no pueden ser adjudicadas a ciertos trabajadores

```
{
= la solucion factible ya no estaria garantizada, es decir, pudiera ser
que el algoritmo no llegue a solucion alguna
~Habria que replantearse el criterio de seleccion para comenzar por
aquellos trabajadores con mas restricciones en cuanto a tareas que no pueden
realizar
~Ya no se garantizaria la solucion optima pero si una factible
}
```

En la solucion al problema de la mochila continua ~~?~~ por que es conveniente la ordenacion previa de los objetos?

{

= Para reducir la complejidad temporal en la toma de cada decision: de  $O(n)$  a  $O(1)$

~ Para reducir la complejidad temporal en la toma de cada decision: de  $O(n^2)$  a  $O(n \log n)$

~ Porque si no se hace no es posible garantizar la toma de decisiones siga un criterio voraz.

}

Supongamos una solucion recursiva que muestra dos caracteristicas: se basa en obtener

soluciones optimas a problemas parciales mas pequeños y por otro, estos subproblemas

se resuelve mas de una vez durante el proceso recursivo

{

~ un algoritmo voraz

= un algoritmo de programacion dinamica

~ un algoritmo del estilo de divide y vencerás

}

La mejora que en general aporta la programacion dinamica frente a la solucion ingenua

se consigue gracias al hecho de que...

{

= ..en la solucion ingenua se resuelve muchas veces un numero relativamente pequeño de

subproblemas distintos.

~ ..en la solucion ingenua se resuelve pocas veces un numero relativamente grande de

subproblemas distintos.

~ El numero de veces que se resuelven los subproblemas no tiene nada que ver con la

eficiencia de los problemas resueltos mediante programacion dinamica

}

De los problemas siguientes... indicad cual no se puede tratar eficientemente como los otros dos

{

- ~ el problema de cortar un tubo
- = el problema de la mochila discreta
- ~ el problema del cambio

}

Un informatico quiere subir una montaña....

{

- ~ un algoritmo divide y vencerás.
- = un algoritmo voraz.
- ~ un algoritmo de programacion dinamica

}

mediante programacion dinamica recursiva unsigned f(unsigned x, unsigned v[] )....

¿Cuál es la mejor estructura para el almacen?

{

- ~ int a[][]
- ~ int a
- = int a[]

}

Dado un problema de optimizacion, el metodo voraz...

{

- ~ ..siempre obtiene la solucion optima
- = Ninguna de las otras dos opciones es cierta

```
~ ..siempre obtiene una solucion factible
}

{
```

Dada la suma de recurrencia....  $T(n)$  sumatorio

```
{
~ pro(n^2)
= pro(2^n)
~ pro($n!$)
}
```

Coeficientes binomiales  $(n r)$ ....

```
{
~ la recursion puede ser infinita y por tanto necesario organizarla segun el
esquema
iterativo de programacion dinamica
~ se repiten muchos calculos y ello se puede evitar haciendo uso de una estrategia
voraz
= se repiten muchos calculos y ello se puede evitar usando programacion dinamica
}
```

En el metodo voraz

```
{
~ ..el dominio de las decisiones solo pueden ser conjuntos discretos o
discretizables
= ..es habitual preparar los datos para disminuir el coste temporal de la funcion
que determina cual es la siguiente decision a tomar
~ ..siempre se encuentra la solucion pero puede que no sea la optima
}
```

El valor que se obtiene con el metodo voraz para el problema de la mochila  
discreta es..

```

{
~ ..una cota inferior para el valor optimo, pero que nunca coincide con este
= ..una cota inferior para el valor optimo que a veces puede ser igual a este
~ ..una cota superior para el valor optimo.

}

```

Dado un problema de optimizacion, el metodo voraz..

```

{
~ ..siempre obtiene la solucion optima
= ..garantiza la solucion optima solo para determinados problemas
~ ..siempre obtiene una solucion factible

}

```

Se pretende implementar mediante programaci?n din?mica iterativa la funci?n recursiva:

```

float f(unsigned x, int y)

if(y < 0) return 0;

float A = 0.0;

if (v1[y] <= x)

A = v2[y] + f(x-v1[y], y-1);

float B = f(x, y-1);

return min(A,2+B);

```

?Mejor estructura para el almacen?

```

{
~ unsigned a[]
~ unsigned a
= unsigned a[][]

}

```

Cuando la descomposicion recursiva de un problema da lugar a subproblemas de tamaño similar ¿que esquema promete ser mas adecuado?

```
{
= programacion dinamica
~ divide y venceras
~ el metodo voraz
}
```

Cual de estos tres problemas de optimizacion no tiene, o no se le conoce, solucion voraz

```
optima
{
~ el arbol de cobertura de coste minimo de un grafo conexo
~ mochila continua
= mochila discreta
}
```

Mediante PD recursiva unsigned f(unsigned x, unsigned v[]) if (x==0) return 0....

¿Mejor complejidad espacial para el almacen?

```
{
~ O(1)
~ O(x^2)
= O(x)
}
```

Un tubo de n centimetros....Di cual de estas tres afirmaciones es falsa

```
{
~ evaluacion exhaustiva de pro(2^n)...
}
```

```
= evaluacion exhaustiva de pro(!n)...
~ evaluacion exhaustiva de j<n...
}
```

Fibonacci

```
{
~ Las dos estrategias serian similares en cuanto a eficiencia
= Programacion Dinamica
~ La estrategia Voraz
}
```

◆Cuál de estas tres estrategias obtiene un mejor valor para la mochila discreta?

```
{
~ Meter primero los elementos de menor peso
~ Meter primero los elementos de mayor peso
= Meter primero los elementos de mayor valor específico o valor por unidad de peso
}
```

El problema de encontrar el arbol de recubrimiento de coste minimo para un grafo no dirigido y ponderado...

```
{
~ ..no se puede resolver en general con una estrategia voraz.
~ solo se puede resolver con una estrategia voraz si existe una arista para cualquier par de vertices del grafo.
= ..se puede resolver siempre con una estrategia voraz.
}
```

La eficiencia de los algoritmos voraces se basa en el hecho de que ...

Seleccione una{

= ... las decisiones tomadas nunca se reconsideran.  
~ ... con antelaci?n, las posibles decisiones se ordenan de mejor a peor.  
~ ... antes de tomar una decisi?n se comprueba si satisface las restricciones del problema.  
}

Si ante un problema de decisi?n existe un criterio de selecci?n voraz entonces ...

Seleccione una:

{  
= Ninguna de las otras dos opciones es cierta.  
~ ... la soluci?n ?ptima est? garantizada.  
~ ... al menos una soluci?n factible est? garantizada.  
}

En la soluci?n al problema de la mochila continua ?por qu? es conveniente la ordenaci?n previa de los objetos?

Seleccione una:

{  
~Porque si no se hace no es posible garantizar que la toma de decisiones siga un criterio voraz.  
~  $O(n^2)$  a  $O(n \log n)$   
=  $O(n)$  a  $O(1)$   
}

Se pretende implementar mediante programaci?n din?mica iterativa la funci?n recursiva:

```
unsigned f(unsigned y, unsigned x) // suponemos y >= x
if (x==0 || y==x) return 1;
return f(y-1, x-1) + f(y-1, x);
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

{  
~O(y^2)  
=O(y)  
~O(1)  
}

¿Cuáles de los siguientes pares de problemas son equivalentes en cuanto al tipo de solución?

(Óptima, factible, etc.) aportada por el método voraz?

Seleccione una:

{  
= El fontanero diligente y la mochila continua.  
~ El fontanero diligente y el problema del cambio.  
~ El fontanero diligente y la asignación de tareas.  
}

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[])
if (x==0)
return 0;

unsigned m = 0;

for (unsigned k = 0; k < x; k++)
m = max(m, v[k] + f(x-k, v));
return m;
```

?

Cuál es la mejor estructura para el almacenamiento?

{  
~ int A  
= int A[]

```
~ int A[][]
}
```

La soluci n de programaci n din mica iterativa del problema de la mochila discreta  
...

Seleccione una:

```
{
```

~ ... tiene un coste temporal asint tico exponencial con respecto al n mero de  
objetos.

~ ... calcula menos veces el valor de la mochila que la correspondiente soluci n  
de

programaci n din mica recursiva.

= ... tiene la restricci n de que los valores tienen que ser enteros positivos.

```
}
```

Se pretende implementar mediante programaci n din mica iterativa la funci n  
recursiva:

```
int f(int x, int y)
if(x <= y) return 1;
return x + f(x-1,y);
```

Qu  es la mejor complejidad espacial que se puede conseguir?

```
{
~ O(x^2)
= O(1)
~ O(x)
}
```

Se pretende implementar mediante programaci n din mica iterativa la funci n  
recursiva:

```
unsigned f(unsigned y, unsigned x) // suponemos y >= x if (x==0 || y==x) return 1;
```

```
return f(y-1, x-1) + f(y-1, x);
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

Seleccione una:

{

~ O(1)

~ O(y^2)

= O(y)

}

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned y, unsigned x) // suponemos y >= x if (x==0 || y==x) return 1;
return f(y-1, x-1) + f(y-1, x);
```

¿Cuál es la mejor estructura para el almacen?

Seleccione una:

{

= int a[][]

~ int a

~ int a[]

}

La programación dinámica...

{

~ ..normalmente se usa para resolver problemas de optimización con dominios discretizables

puesto que las tablas se han de indexar con este tipo de valores

~ ..en algunos casos se puede utilizar para resolver problemas de optimización con dominios

continuos pero probablemente pierda su eficacia ya que puede disminuir drásticamente

el número de subproblemas repetidos.

= Las otras dos opciones son correctas.

}

Se pretende implementar mediante programaci?n din?mica iterativa la funci?n recursiva:

```
float f(unsigned x, int y)
if(y < 0) return 0;
float A = 0.0;
if (v1[y] <= x)
A = v2[y] + f(x-v1[y], y-1);
float B = f(x, y-1);
return min(A,2+B);
```

?Cu? es la mejor complejidad espacial que se puede conseguir?{

~ O(1)

= O(y)

~ O(y^2)

}

?Cu? de los siguientes pares de problemas son equivalentes en cuanto al tipo de soluci?n

(?ptima, factible, etc.) aportada por el m?todo voraz?

Seleccione una:

{

= La mochila discreta y la asignaci?n de tareas.

~ La mochila continua y la asignaci?n de tareas.

~ El fontanero diligente y el problema del cambio.

}

Los algoritmos de programacion dinamica hacen uso..

{

~ ..de que la solucion optima se puede construir añadiendo a la solucion el elemento

optimo de los elementos restantes, uno a uno.

= ..de que se puede ahorrar calculos guardando resultados anteriores en un almacen.

~ ..de una estrategia trivial consistente en examinar todas las soluciones posibles.

}

**Pregunta 1**

Correcta

Puntúa como 1,00

 Marcar pregunta

**En los algoritmos de ramificación y poda ...**

Seleccione una:

- a. Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.
- b. El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida.
- c. Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima. ✓

**Pregunta 2**

Incorrecta

Puntúa como 1,00

 Marcar pregunta

Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila:

Seleccione una:

- a. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido. **X**
- b. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objeto.
- c. El valor óptimo de la mochila continua correspondiente.

**Pregunta 3**

**Sin contestar**

**Puntúa como 1,00**

 **Marcar pregunta**

Cuando se resuelve usando un algoritmo de ramificación y poda un problema de  $n$  decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?

**Seleccione una:**

a.  $O(2^n)$

b.  $O(n!)$

c.  $O(n^2)$

#### Pregunta 4

Incorrecta

Puntúa como 1,00

 Marcar pregunta

En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento ...

Seleccione una:

- a. ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- b. ... pueden eliminar soluciones parciales que son factibles.
- c. Las dos anteriores son verdaderas. 

**Pregunta 5**

Correcta

Puntúa como 1,00

 Marcar pregunta

Se desea obtener todas las permutaciones de una lista compuesta por  $n$  elementos. ¿Qué esquema es el más adecuado?

Seleccione una:



a. Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.



b. Ramificación y poda, puesto que con buenas funciones de cota es más eficiente para este problema que vuelta atrás.



c. Vuelta atrás, para este problema no hay un esquema más eficiente. ✓

**Pregunta 6**

Correcta

Puntuía como 1,00

 Marcar pregunta

La complejidad en el peor de los casos de un algoritmo de vuelta atrás ...

Seleccione una:

- a. ... es exponencial con el número de decisiones a tomar. ✓
- b. ... puede ser exponencial con el número de alternativas por cada decisión.
- c. ... puede ser polinómica con el número de decisiones a tomar.

**Pregunta 7**

Sin contestar

Puntúa como 1,00

 Marcar pregunta

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

Seleccione una:

- a. Sería una cota optimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.
- b. Sería una cota pesimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.
- c. Ninguna de las otras dos opciones.

**Pregunta 8**

Correcta

Puntúa como 1,00



¿Cuál es la diferencia principal entre una solución de *vuelta atrás* y una solución de *ramificación y poda* para el problema de la mochila?

Seleccione una:

- 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. ✓

**Pregunta 9**

Correcta

Puntúa como 1,00

 Marcar pregunta

La estrategia de *ramificación y poda* genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones. ✓
- b. ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- c. ... un recorrido en anchura del árbol que representa el espacio de soluciones.

**Pregunta 10**

Incorrecta

Puntúa como 1,00



Marcar pregunta

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir. X
- b. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
- c. Sí, siempre es así.

**Pregunta 11**

Correcta

Puntúa como 1,00

 Marcar pregunta

El uso de funciones de cota en ramificación y poda ...

Seleccione una:

- a. ... puede reducir el número de instancias del problema que pertenecen al caso peor. ✓
- b. ... transforma en polinómicas complejidades que antes eran exponenciales.
- c. ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.

## Pregunta 12

Sin contestar

Puntuación 1,00

 Marcar pregunta

Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

Seleccione una:

- 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 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.
- c. Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.

Pregunta 1

Correcta

Puntúa como 1,00

→ Marcar pregunta

En los algoritmos de **ramificación y poda** ...

Seleccione una:

- 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 pesimista es el beneficio esperado de cualquier nodo factible que no es el óptimo.
- c. Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima. ✓

**Pregunta 2**

Sin contestar

Puntúa como 1,00

 **Marcar pregunta**

Al resolver el problema del viajante de comercio mediante **vuelta atrás** y asumiendo un grafo de  $n$  vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

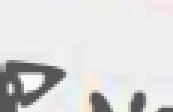
Seleccione una:

- a. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
- b. Se multiplica  $n$  por la distancia de la arista más corta que nos queda por considerar.
- c. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las  $n$  aristas más cortas.

Pregunta 3

Correcta

Puntúa como 1,00

 **Marcar**

preguntas

¿Cuál es la diferencia principal entre una solución de vuelta atrás y una solución de ramificación y poda para el problema de la mochila?

Seleccione una:

- 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 orden de exploración de las soluciones. ✓
- c. El coste asintótico en el caso peor.

Pregunta 4

Correcta

Puntúa como 1,00

 Marcar

Pregunta

274,6 x 168,5 mm

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de la mochila continua correspondiente.
- b. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos. ✓
- c. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.

**Pregunta 5**

Sin contestar

Puntúa como 1.00

 Marcar pregunta

La complejidad en el mejor de los casos de un algoritmo de vuelta atrás ...

Seleccione una:

- a. ... puede ser polinómica con el número de decisiones a tomar.
- b. ... suele ser polinómica con el número de alternativas por cada decisión
- c. ... es siempre exponencial con el número de decisiones a tomar.

**Pregunta 6**

Sin contestar

Puntúa como 1,00

🚩 Marcar pregunta

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

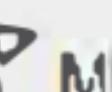
Seleccione una:

- a. Ninguna de las otras dos opciones.
- b. Sería una cota pesimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.
- c. Sería una cota optimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.

Pregunta 7

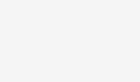
Incorrecta

Puntúa como 1.00

 Marcar pregunta

Tratándose de un problema de optimización, en la lista de nodos vivos de *ramificación y poda* ...

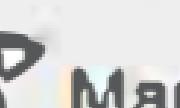
Seleccione una:

- a. ... sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento.
- b. Las otras dos opciones son ciertas.
- c. ... puede haber nodos que no son prometedores. 

**Pregunta 8**

Incorrecta

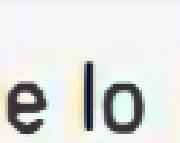
Puntúa como 1.00

 Marcar

pregunta

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

Seleccione una:

- a. una estrategia voraz puede ser la única alternativa.
- b. podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.
- c. es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione. 

Pregunta 1

Incorrecta

Puntúa como 1.00

 Marcar pregunta

La complejidad en el mejor de los casos de un algoritmo de ramificación y poda...

Seleccione una:

- a. ... es siempre exponencial con el número de decisiones a tomar. 
- b. ... suele ser polinómica con el número de alternativas por cada decisión
- c. ... puede ser polinómica con el número de decisiones a tomar.

**Pregunta 2**

Correcta

Puntúa como 1.00

 **Marcar pregunta**

La estrategia de *ramificación y poda* genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido en profundidad del árbol que representa el 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 anchura del árbol que representa el espacio de soluciones.

### Pregunta 3

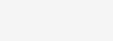
Correcto

Puntúa como 1,00

 [Marcar pregunta](#)

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

Seleccione una:

- a. podremos aplicar el esquema **vuelta atrás** siempre que se trate de un conjunto **infinito numerable**.
- b. es probable que a través de **programación dinámica** se obtenga un algoritmo eficaz que lo **solucione**.
- c. una estrategia **voraz** puede ser la **única alternativa**. 

Pregunta 4

Correcta

Puntúa como 1.00

 Marcar pregunta

## El uso de funciones de cota en ramificación y poda ...

Seleccione una:

- 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. ✓

**Pregunta 5**

[Sin contestar](#)

Puntaje como 1.00

 [Marcar pregunta](#)

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?

Seleccione una:

- a. La comprobación de las soluciones factibles: en *ramificación y poda* no es necesario puesto que sólo genera nodos factibles.
- b. Aprovechamos mejor las cotas optimistas.
- c. Cambiamos la función que damos a la cota pesimista.

**Pregunta 6**

Sin contestar

Puntuación 1,00

 Marcar pregunta

El problema de cortar un tubo de longitud  $n$  en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...

Seleccione una:

- a. ... no se puede resolver usando un algoritmo de vuelta atrás.
- b. ... se debe resolver mediante un algoritmo de vuelta atrás, dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo.
- c. ... se puede resolver mediante un algoritmo de vuelta atrás pero existe una solución asintóticamente mucho más eficiente.

Pregunta 7

Si no contestar

Puntúa como 1.00

 Marcar pregunta

En los algoritmos de **ramificación y poda**, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
- b. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
- c. Sí, siempre es así.

Pregunta 8

Correcta

Puntúa como 1.00

 Marcar pregunta

Cuando se resuelve usando un algoritmo de ramificación y poda un problema de  $n$  decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a.  $O(2^n)$  ✓
- b.  $O(n^2)$
- c.  $O(n!)$

Pregunta 9

Si no contestar

Puntúa como 1,00

 Marcar pregunta

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

Seleccione una:

- a. Ninguna de las otras dos opciones.
- b. Sería una cota pesimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.
- c. Sería una cota optimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.

Pregunta 10

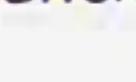
Correcta

Puntúa como 1,00

 Marcar pregunta

Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

Seleccione una:

- 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. 

Pregunta 11

Correcta

Puntúa como 1,00

 Marcar pregunta

En los algoritmos de *ramificación y poda* ...

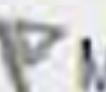
Seleccione una:

- a. Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima. ✓
- b. Una cota pesimista es necesariamente un valor inalcanzable, de no ser así no está garantizado que no se eliminen nodos factibles.
- c. Una cota pesimista es el beneficio esperado de cualquier nodo factible.

Pregunta 12

Correcta

Puntúa como 1,00

 Marcar pregunta

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- b. El valor de la mochila continua correspondiente.
- c. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.



**Pregunta 9** El problema de cortar un tubo de longitud  $n$  en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...

**Correcta**

Puntúa como 1.00

Marcar pregunta

Seleccione una:

- a. ... no se puede resolver usando un algoritmo de *vuelta atrás*.
- b. ... se puede resolver mediante un algoritmo de *vuelta atrás* pero existe una solución asintóticamente mucho más eficiente. ✓
- c. ... se debe resolver mediante un algoritmo de *vuelta atrás*, dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo.

**Pregunta 10**

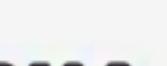
Incorrecta

Puntúa como 1,00

 Marcar pregunta

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir. 
- b. Sí, siempre es así.
- c. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.

**Pregunta 11**

Sin contestar

Puntúa como 1.00

 Marcar

pregunta

En la estrategia de *ramificación y poda* ...

Seleccione una:

a. ... cada nodo tiene su propia cota pesimista y también su propia cota optimista.

b. ... cada nodo tiene su propia cota optimista, la cota pesimista sin embargo, es común para todos los nodos.

c. ... cada nodo tiene su propia cota pesimista, la cota optimista sin embargo, es común para todos los nodos.

Pregunta 12

Corrección

Puntúa como 1.00

 Marcar pregunta

Cuando se resuelve usando un algoritmo de ramificación y poda un problema de  $n$  decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a.  $O(2^n)$  ✓
- b.  $O(n^2)$
- c.  $O(n!)$

En ausencia de cotas optimistas y pesimistas, la estrategia de *vuelta atrás* ...

Seleccione una:

- a. ... no se puede usar para resolver problemas de optimización.
- b. ... debe recorrer siempre todo el árbol.
- c. ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles. ✓

En el esquema de *vuelta atrás*, los mecanismos de poda basados en la mejor solución hasta el momento ...

Seleccione una:

- a. Las dos anteriores son verdaderas.
- b. ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- c. ... pueden eliminar soluciones parciales que son factibles. ✓

Si para resolver un **mismo problema** usamos un algoritmo de *ramificación y poda* y lo modificamos mínimamente para convertirlo en un algoritmo de *vuelta atrás*, ¿qué cambiamos realmente?

Seleccione una:

- a. Cambiamos la función que damos a la cota pesimista.
- b. Provocamos que las cotas optimistas pierdan eficacia.
- c. Sería necesario comprobar si las soluciones son factibles o no puesto que *ramificación y poda* sólo genera nodos factibles.

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?

Seleccione una:

- a. La comprobación de las soluciones factibles: en *ramificación y poda* no es necesario puesto que sólo genera nodos factibles. 
- b. Cambiamos la función que damos a la cota pesimista.
- c. Aprovechamos mejor las cotas optimistas.

En la estrategia de *ramificación y poda* ...

Seleccione una:

- a. ... cada nodo tiene su propia cota pesimista y también su propia cota optimista. ✓
- b. ... cada nodo tiene su propia cota pesimista, la cota optimista sin embargo, es común para todos los nodos.
- c. ... cada nodo tiene su propia cota optimista, la cota pesimista sin embargo, es común para todos los nodos.

La estrategia de ramificación y poda necesita cotas pesimistas ...

Seleccione una:

- a. ... para decidir el orden de visita de los nodos del árbol de soluciones
- b. ... sólo si se usa para resolver problemas de optimización. ✓
- c. ... para determinar si una solución es factible

Cuando resolvemos un problema mediante un esquema de ramificación y poda ...

Seleccione una:

- a. ... los valores entre los cuales se elige en cada una de las decisiones tienen que formar un conjunto finito. ✓
- b. ... las decisiones sólo pueden ser binarias.
- c. ... los valores entre los cuales se elige en cada una de las decisiones pueden formar un conjunto infinito.

Tratándose de un problema de optimización, en la lista de nodos vivos de *ramificación y poda* ...

Seleccione una:

- a. ... sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento.
- b. ... puede haber nodos que no son prometedores.
- c. Las otras dos opciones son ciertas. ✓

¿Cuál es la diferencia principal entre una solución de *vuelta atrás* y una solución de *ramificación y poda* para el problema de la mochila?

Seleccione una:

- a. El orden de exploración de las soluciones. ✓
- b. El coste asintótico en el caso peor.
- c. El hecho que la solución de *ramificación y poda* puede empezar con una solución subóptima voraz y la de *vuelta atrás* no.

La ventaja de la estrategia *ramificación y poda* frente a *vuelta atrás* es que la primera genera las soluciones **posibles** al problema mediante ...

Seleccione una:

- a. ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.
- b. Las otras dos opciones son verdaderas. ✓
- c. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.

En los algoritmos de *ramificación y poda* ...

Seleccione una:

- a. El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida.
- 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 optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.

En los algoritmos de **ramificación y poda**, ¿el valor de una cota pesimista es menor que el valor de una cota optimista?  
(entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
- b. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir. ✓
- c. Sí, siempre es así.

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- 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, nunca es así.

¿Para qué sirven las cotas pesimistas en ramificación y poda?

Seleccione una:

- a. Para tener la certeza de que la cota optimista está bien calculada.
- b. Para descartar nodos basándose en la preferencia por algún otro nodo ya completado.
- c. Para descartar nodos basándose en el beneficio esperado. ✓

La estrategia de *ramificación y poda* genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones. ✓
- b. ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- c. ... un recorrido en anchura del árbol que representa el espacio de soluciones.

La complejidad en el peor de los casos de un algoritmo de ramificación y poda ...

Seleccione una:

- a ... puede ser exponencial con el número de alternativas por cada decisión.
- b ... puede ser polinómica con el número de decisiones a tomar.
- c ... es exponencial con el número de decisiones a tomar. ✓

La complejidad en el mejor de los casos de un algoritmo de ramificación y poda...

Seleccione una:

- a. ... es siempre exponencial con el número de decisiones a tomar.
- b. ... suele ser polinómica con el número de alternativas por cada decisión
- c. ... puede ser polinómica con el número de decisiones a tomar. ✓

Se desea obtener todas las permutaciones de una lista compuesta por  $n$  elementos. ¿Qué esquema es el más adecuado?

Seleccione una:

- a. Ramificación y poda, puesto que con buenas funciones de cota es más eficiente para este problema que vuelta atrás.
- b. Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
- c. Vuelta atrás, para este problema no hay un esquema más eficiente. ✓

Al resolver el problema del viajante de comercio mediante vuelta atrás y asumiendo un grafo de  $n$  vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Seleccione una:

- a. Se multiplica  $n$  por la distancia de la arista más corta que nos queda por considerar.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las  $n$  aristas más cortas.
- c. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido. ✓

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 en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a.  $O(2^n)$  ✓
- b.  $O(n!)$
- c.  $O(n^2)$

## El uso de funciones de cota en ramificación y poda ...

Seleccione una:

- a. ... transforma en polinómicas complejidades que antes eran exponenciales.
- b. ... puede reducir el número de instancias del problema que pertenecen al caso peor. ✓
- c. ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.

Di cuál de estas tres soluciones a problemas de optimización no comporta, en el peor caso, tener que considerar  $O(n!)$  posibilidades.

Seleccione una:

- a. La solución de *vuelta atrás* al problema del viajante de comercio (*travelling salesman problem*), o sea, el de encontrar un ciclo hamiltoniano de coste mínimo en un grafo conexo de  $n$  vértices donde cada arista tiene un coste asignado. 
- b. La solución de *ramificación y poda* al problema de la asignación de  $n$  tareas a  $n$  trabajadores de forma que cada trabajador hace exactamente una tarea y cada tarea es asignada a un trabajador exactamente, de forma que la suma de los costes de las tareas es mínimo. 
- 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*).

Al resolver el problema del viajante de comercio mediante *vuelta atrás*, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

Seleccione una:

- 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. 

Al resolver el problema del viajante de comercio mediante vuelta atrás y asumiendo un grafo de  $n$  vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Seleccione una:

- a. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las  $n$  aristas más cortas.
- c. Se multiplica  $n$  por la distancia de la arista más corta que nos queda por considerar.

El problema de cortar un tubo de longitud  $n$  en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...

Seleccione una:

- a. ... no se puede resolver usando un algoritmo de *vuelta atrás*.
- b. ... se puede resolver mediante un algoritmo de *vuelta atrás* pero existe una solución asintóticamente mucho más eficiente. ✓
- c. ... se debe resolver mediante un algoritmo de *vuelta atrás*, dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en linea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad de entre las posibles según el mapa de carreteras, que esté más cercana al destino según su distancia geográfica. Este algoritmo voraz ¿semiria como cota pesimista?

Seleccione una:

- a. No, ya que no asegura que se encuentre una solución factible. ✓
- b. Sí, puesto que la distancia geográfica asegura que otra solución mejor no es posible.
- c. No, ya que en algunos casos puede dar distancias menores que la óptima.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, \$-1\$) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. 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.

Seleccione una:

- a. El nuevo algoritmo solo será más rápido para algunas instancias del problema ✓
- b. Esta estrategia no asegura que se obtenga el camino mas corto.
- c. El nuevo algoritmo siempre sea más rápido.

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 linea recta) entre cada par de ciudades para como cota para limitar la búsqueda en un algoritmo de vuelta atrás.

¿Qué tipo de cota seria?

Seleccione una:

- a. No se trataría de ninguna poda puesto que es posible que esa heurística no encuentre una solución factible.
- b. Una cota pesimista.
- c. Una cota optimista. ✓

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en linea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en linea recta.

¿Qué tipo de cota sería?

Seleccione una:

- a. Sería una cota pesimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible. ✓
- b. Ninguna de las otras dos opciones.
- c. Sería una cota optimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.

Dado un problema de optimización cualquiera, ¿la estrategia de vuelta atrás garantiza la solución óptima?

Seleccione una:

- a. Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.
- b. Sí, puesto que ese método analiza todas las posibilidades.
- c. 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.

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

Seleccione una:

- a. podremos aplicar el esquema *vuelta atrás* siempre que se trate de un conjunto infinito numerable.
- b. es probable que a través de *programación dinámica* se obtenga un algoritmo eficaz que lo solucione.
- c. una estrategia voraz puede ser la única alternativa. ✓

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor de la mochila continua correspondiente.
- c. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.

Decid cuál de estas tres no sirve como cota optimista para obtener el valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- b. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos. ✓
- c. El valor de la mochila continua correspondiente.

Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila.

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- c. El valor óptimo de la mochila continua correspondiente.

La estrategia de *vuelta atrás* es aplicable a problemas de selección y optimización en los que:

Seleccione una:

- a. El espacio de soluciones puede ser tanto finito como infinito pero en este último caso debe ser al menos numerable.
- b. El espacio de soluciones es un conjunto infinito.
- c. El espacio de soluciones es un conjunto finito. ✓

**Pregunta 12**

Respuesta  
guardada

Puntúa como 1.00

 Marcar  
pregunta

Dado un problema de optimización cualquiera. ¿la estrategia de vuelta atrás garantiza la solución óptima?

Seleccione una:

- a. Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.
- b. Sí, puesto que ese método analiza todas las posibilidades.
- c. 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.

**Pregunta 11**

Respuesta  
guardada

Puntúa como 1.00

 **Marcas**  
pregunta

¿Cuál es la diferencia principal entre una solución de *vuelta atrás* y una solución de *ramificación y poda* para el problema de la mochila?

Seleccione una:

a. El orden de exploración de las soluciones.

b. El hecho que la solución de *ramificación y poda* puede empezar con una solución subóptima voraz y la de *vuelta atrás* no.

c. El coste asintótico en el caso peor.

**Pregunta 10**

Sin responder aún

Puntúa como 1.00

 Marcar pregunta

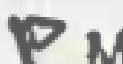
**La complejidad en el mejor de los casos de un algoritmo de ramificación y poda...**

**Seleccione una:**

- a. ... es siempre exponencial con el número de decisiones a tomar.
- b. ... suele ser polinómica con el número de alternativas por cada decisión
- c. ... puede ser polinómica con el número de decisiones a tomar.

**Pregunta 9**Respuesta  
guardada

Puntúa como 1.00

 **Marcar  
pregunta**

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 en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a.  $O(n!)$
- b.  $O(n^2)$
- c.  $O(2^n)$

**Pregunta 8**

Respuesta  
guardada

Puntúa como 1.00

 Marcar  
pregunta

## El uso de funciones de cota en ramificación y poda ...

Seleccione una:

- a. ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.
- b. ... transforma en polinómicas complejidades que antes eran exponenciales.
- c. ... puede reducir el número de instancias del problema que pertenecen al caso peor.

Pregunta 7

Respuesta  
guardada

Puntúa como 1,00

► Marcar  
pregunta

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta arrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino según su distancia geográfica.

Este algoritmo voraz. ¿serviría como cota pesimista?

Seleccione una:

- a. No, ya que en algunos casos puede dar distancias menores que la óptima.
- b. No, ya que no asegura que se encuentre una solución factible.
- c. Sí, puesto que la distancia geográfica asegura que otra solución mejor no es posible.

## Pregunta 6

Respuesta  
guardada

Puntúa como 1.00

 Marcar  
pregunta

Se desea obtener todas las permutaciones de una lista compuesta por  $N$  elementos. ¿Qué esquema es el más adecuado?

Seleccione una:



- a. *Ramificación y poda*, puesto que con buenas funciones de cota es más eficiente para este problema que *vuelta atrás*.
- b. *Vuelta atrás*, para este problema no hay un esquema más eficiente.
- c. *Divide y vencerás*, puesto que la división en sublistas se podría hacer en tiempo constante.

**Pregunta 5**

Respuesta  
guardada

Puntúa como 1,00

 Marcar  
pregunta

En los algoritmos de *ramificación y poda* ...

Seleccione una:

a. Una cota pesimista es el beneficio esperado de cualquier nodo factible que no es el óptimo.

b. Una cota optimista es necesariamente un valor alcanzable. de no ser así no está garantizado que se encuentre la solución óptima.

c. Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.

**Pregunta 3**

**Respuesta  
guardada**

**Puntúa como 1.00**

**M**arcar  
pregunta

Decíd cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- b. El valor de la mochila continua correspondiente.
- c. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

**Pregunta 2**

Resuesta  
guardada

Puntuación 1,00

 **Marcar  
pregunta**

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. No, nunca es así.
- b. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
- c. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.

**Pregunta 1**

**Respuesta  
guardada**

**Puntúa como 1.00**

 **Marcar  
pregunta**

**La estrategia de ramificación y poda genera las soluciones posibles al problema mediante ...**

**Seleccione una:**

- a. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.
- b. ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- c. ... un recorrido en anchura del árbol que representa el espacio de soluciones.

**Pregunta 12**

Correcta

Puntúa como 1,00

 Marcar

 Pregunta

En los algoritmos de **ramificación y poda**, ¿el valor de una cota pesimista es **mayor que el valor de una cota optimista**? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- 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, nunca es así.

Pregunta 11

Correcta

Puntúa como 1.00

 Marcar

pregunta

La complejidad en el peor de los casos de un algoritmo de ramificación y poda ...

Seleccione una:

- a. ... puede ser exponencial con el número de alternativas por cada decisión.
- b. ... es exponencial con el número de decisiones a tomar. ✓
- c. ... puede ser polinómica con el número de decisiones a tomar.

**Pregunta 9**

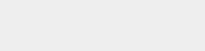
Incorrecta

Puntúa como 1,00

 Marcar pregunta

Cuando se resuelve usando un algoritmo de ramificación y poda un problema de  $n$  decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a.  $O(2^n)$
- b.  $O(n!)$  
- c.  $O(n^2)$

Pregunta 8

Correcto

Puntúa como 1.00

 Marcar pregunta

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en linea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

Seleccione una:

- a. Sería una cota pesimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible. ✓
- b. Sería una cota optimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.
- c. Ninguna de las otras dos opciones.

Pregunta 7

Sin contestar

Puntúa como 1.00

 Marcar

Preguntas

El problema de cortar un tubo de longitud  $n$  en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...

Seleccione una:

- a. ... se debe resolver mediante un algoritmo de vuelta atrás, dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo.
- b. ... se puede resolver mediante un algoritmo de vuelta atrás pero existe una solución asintóticamente mucho más eficiente.
- c. ... no se puede resolver usando un algoritmo de vuelta atrás.

**Pregunta 6**

Correcta

Puntúa como 1.00

 Marcar

Pregunta

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

Seleccione una:

- 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. ✓

**Pregunta 4**

Correcto

Puntúa como 1.00

 Marcar

Preguntas

La estrategia de *ramificación y poda* genera las soluciones posibles al problema mediante ...

Seleccione una:

a. ... un recorrido en anchura del árbol que representa el 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.

**Pregunta 3**

Correcta

Puntúa como 1.00

 Marcar

 Pregunta

En los algoritmos de *ramificación y poda* ...

Seleccione una:

- a. Una cota pesimista es el beneficio esperado de cualquier nodo factible que no es el óptimo.
- 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 optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.

**Pregunta 2**

Correcto

Puntúa como 1.00

 Marcar pregunta

La ventaja de la estrategia ramificación y poda frente a vuelta atrás es que la primera genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. Las otras dos opciones son verdaderas. ✓
- b. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.
- c. ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.

Pregunta 1

Sin contestar

Puntúa como 1.00

Marcar

Pregunta

Al resolver el problema del viajante de comercio mediante *vuelta atrás*, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

Seleccione una:

- 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 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.
- c. Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.

**Pregunta 12**

**Sin contestar**

**Puntúa como 1.00**

 **Marcar pregunta**

**El valor que se obtiene con el método voraz para el problema de la mochila discreta es ...**

**Seleccione una:**

- a. ... una cota inferior para el valor óptimo que a veces puede ser igual a este.
- b. ... una cota inferior para el valor óptimo, pero que nunca coincide con este.
- c. ... una cota superior para el valor óptimo.

Pregunta 11  
Correcta

Puntúa como 1.00

 Marcar pregunta

Cuando se calculan los coeficientes binomiales usando la recursión  $\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r}$ , con  $\binom{n}{0} = \binom{n}{n} = 1$ , qué problema se da y cómo se puede resolver?

Seleccione una:

- a. Se repiten muchos cálculos y ello se puede evitar usando programación dinámica. ✓
- b. La recursión puede ser infinita y por tanto es necesario organizarla según el esquema iterativo de programación dinámica.
- c. Se repiten muchos cálculos y ello se puede evitar haciendo uso de una estrategia voraz.

Pregunta 10

Correcto

Puntaje como 1.00



Marcar pregunta

am

¿Cuál de estas tres estrategias voraces obtiene un mejor valor para la mochila discreta?

Seleccione una:

- a. Meter primero los elementos de mayor valor específico o valor por unidad de peso. ✓
- b. Meter primero los elementos de menor peso.
- c. Meter primero los elementos de mayor valor.

### Pregunta 9

Sin contestar

Puntúa como 1.00



Preguntas

De los problemas siguientes, indicad cuál no se puede tratar eficientemente como los otros dos:

Seleccione una:

- 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 el mínimo de monedas posibles.
- c. El problema de la mochila sin fraccionamiento y sin restricciones en cuanto al dominio de los pesos de los objetos y de sus valores.

Pregunta 8

Incorrecta

Puntúa como 1.00

Marcar pregunta

Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
unsigned f(unsigned x, unsigned v[]) {
 if (x==0)
 return 0;
 unsigned m = 0;
 for (unsigned k = 0; k < x; k++)
 m = max(m, v[k] + f(x-k, v));
 return m;
}
```

¿Cuál es la mejor estructura para el almacén?

Seleccione una:

a. int A[][]

b. int A[]

c. int A

**Pregunta 10**

Correcto

Puntúa como 1.00

 Marcar

Pregunta

En la estrategia de *ramificación y poda* ...

Seleccione una:

- a. ... cada nodo tiene su propia cota pesimista, la cota optimista sin embargo, es común para todos los nodos.
- b. ... cada nodo tiene su propia cota optimista, la cota pesimista sin embargo, es común para todos los nodos.
- c. ... cada nodo tiene su propia cota pesimista y también su propia cota optimista. ✓

Pregunta 1  
Correcta  
Puntúa como 1,00  
 Marcar pregunta

Se desea obtener todas las permutaciones de una lista compuesta por  $n$  elementos. ¿Qué esquema es el más adecuado?

Seleccione una:

- a. Ramificación y poda, puesto que con buenas funciones de cota es más eficiente para este problema que vuelta atrás.
- b. Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
- c. Vuelta atrás, para este problema no hay un esquema más eficiente. ✓

Pregunta 2  
Sin contestar  
Puntúa como 1,00  
 Marcar pregunta

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor de la mochila continua correspondiente.
- c. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.

Pregunta 3  
Incorrecta  
Puntúa como 1,00  
 Marcar pregunta

Dado un problema de optimización cualquiera, ¿la estrategia de vuelta atrás garantiza la solución óptima?

Seleccione una:

- 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.

Pregunta 4  
Correcta  
Puntúa como 1,00  
 Marcar pregunta

La estrategia de ramificación y poda genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- b. ... un recorrido en anchura del árbol que representa el espacio de soluciones.
- c. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones. ✓

Pregunta 5  
Incorrecta  
Puntúa como 1,00  
 Marcar pregunta

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?

Seleccione una:

- a. Cambiamos la función que damos a la cota pesimista.
- b. Aprovechamos mejor las cotas optimistas.
- c. La comprobación de las soluciones factibles: en ramificación y poda no es necesario puesto que sólo genera nodos factibles. ✗

Pregunta 6  
Correcta  
Puntúa como 1,00  
 Marcar pregunta

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 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 geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

Seleccione una:

- a. Sería una **cota pesimista** siempre que se tenga la certeza de que esa aproximación encuentra una solución factible. ✓
- b. Ninguna de las otras dos opciones.
- c. Sería una **cota optimista** siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.

Pregunta 7  
Correcta  
Puntúa como 1,00  
 Marcar pregunta

¿Para qué sirven las cotas pesimistas en ramificación y poda?

Seleccione una:

- a. Para tener la certeza de que la cota optimista está bien calculada.
- b. Para descartar nodos basándose en la preferencia por algún otro nodo ya completado.
- c. Para descartar nodos basándose en el beneficio esperado. ✓

Pregunta 8  
Incorrecta  
Puntúa como 1,00  
 Marcar pregunta

Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

Seleccione una:

- a. Se multiplica  $k_C$  por la distancia de la arista más corta que nos queda por considerar, donde  $k_C$  es el número de saltos que nos quedan por dar.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las  $k_C$  aristas más cortas, donde  $k_C$  es el número de saltos que nos quedan por dar.
- c. Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido. ✗

**Pregunta 9**

Correcta

Puntúa como 1,00

Marcar pregunta

La complejidad en el peor de los casos de un algoritmo de *ramificación y poda* ...

Seleccione una:

- a. ... puede ser exponencial con el número de alternativas por cada decisión.
- b. ... puede ser polinómica con el número de decisiones a tomar.
- c. ... es exponencial con el número de decisiones a tomar.

**Pregunta 10**

Incorrecta

Puntúa como 1,00

Marcar pregunta

En los algoritmos de *ramificación y poda* ...

Seleccione una:

- a. Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.
- b. El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida.
- c. Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.

**Pregunta 11**

Sin contestar

Puntúa como 1,00

Marcar pregunta

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
- b. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
- c. Sí, siempre es así.

**Pregunta 12**

Sin contestar

Puntúa como 1,00

Marcar pregunta

Cuando se resuelve usando un algoritmo de ramificación y poda un problema de  $\eta$  decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a.  $O(\eta!)$
- b.  $O(2^\eta)$
- c.  $O(\eta^2)$

**Comenzado el** martes, 21 de mayo de 2013, 11:20

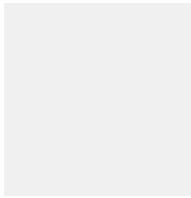
**Complegado el** martes, 21 de mayo de 2013, 11:51

**Tiempo empleado** 30 minutos 11 segundos

## Pregunta 1

Correcta

Puntúa como 1,00

 Marcar pregunta

### Texto de la pregunta

La ventaja de la estrategia *ramificación y poda* frente a *vuelta atrás* es que la primera genera las soluciones posibles al problema mediante ...

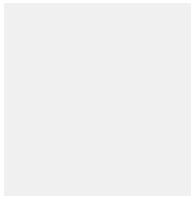
Seleccione una:

- a. ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.
- b. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.
- c. Las otras dos opciones son verdaderas. ✓

## Pregunta 2

Incorrecta

Puntúa como 1,00

 Marcar pregunta

### Texto de la pregunta

En los algoritmos de *ramificación y poda* ...

Seleccione una:

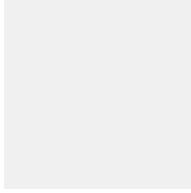
- a. El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida.
- b. Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.

- c. Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima. X

## Pregunta 3

Incorrecta

Puntúa como 1,00

 Marcar pregunta

### Texto de la pregunta

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo *voraz* basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino según su distancia geográfica.

Este algoritmo voraz, ¿serviría como cota pesimista?

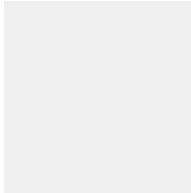
Seleccione una:

- a. No, ya que en algunos casos puede dar distancias menores que la óptima.
- b. Sí, puesto que la distancia geográfica asegura que otra solución mejor no es posible. X
- c. No, ya que no asegura que se encuentre una solución factible.

## Pregunta 4

Sin contestar

Puntúa como 1,00

 Marcar pregunta

### Texto de la pregunta

Di cuál de estas tres soluciones a problemas de optimización no comporta, en el peor caso, tener que considerar  $O(n!)$  posibilidades.

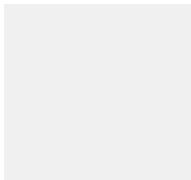
Seleccione una:

- a. La solución de *vuelta atrás* al problema del viajante de comercio (*travelling salesman problem*), o sea, el de encontrar un ciclo hamiltoniano de coste mínimo en un grafo conexo de  $n$  vértices donde cada arista tiene un coste asignado.
- b. 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*).
- c. La solución de *ramificación y poda* al problema de la asignación de  $n$  tareas a  $n$  trabajadores de forma que cada trabajador hace exactamente una tarea y cada tarea es asignada a un trabajador exactamente, de forma que la suma de los costes de las tareas es mínimo.

## Pregunta 5

Correcta

Puntúa como 1,00

 Marcar pregunta

### Texto de la pregunta

Al resolver el problema del viajante de comercio mediante *vuelta atrás*, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

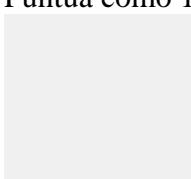
Seleccione una:

- a. Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
- b. 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.
- 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. ✓

## Pregunta 6

Sin contestar

Puntúa como 1,00

 Marcar pregunta

### Texto de la pregunta

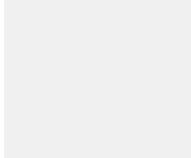
La complejidad en el mejor de los casos de un algoritmo de *ramificación y poda*...  
Seleccione una:

- a. ... puede ser polinómica con el número de decisiones a tomar.
- b. ... es siempre exponencial con el número de decisiones a tomar.
- c. ... suele ser polinómica con el número de alternativas por cada decisión

## Pregunta 7

Correcta

Puntúa como 1,00

 Marcar pregunta

### Texto de la pregunta

Tratándose de un problema de optimización, en la lista de nodos vivos de *ramificación y poda* ...

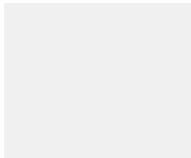
Seleccione una:

- a. ... sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento.
- b. ... puede haber nodos que no son prometedores.
- c. Las otras dos opciones son ciertas. ✓

## Pregunta 8

Incorrecta

Puntúa como 1,00

 Marcar pregunta

### Texto de la pregunta

Dado un problema de optimización cualquiera, ¿la estrategia de *vuelta atrás* garantiza la solución óptima?

Seleccione una:

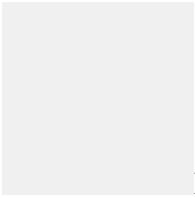
- a. Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.

- b. Sí, puesto que ese método analiza todas las posibilidades. X
- c. 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.

## Pregunta 9

Correcta

Puntúa como 1,00

 Marcar pregunta

### Texto de la pregunta

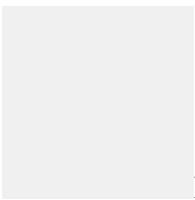
Cuando se resuelve usando un algoritmo de ramificación y poda un problema de  $n$  decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar? Seleccione una:

- a.  $O(n!)$
- b.  $O(n^2)$
- c.  $O(2^n)$  ✓

## Pregunta 10

Correcta

Puntúa como 1,00

 Marcar pregunta

### Texto de la pregunta

Decid cuál de estas tres no sirve como cota optimista para obtener el valor óptimo de la mochila discreta:

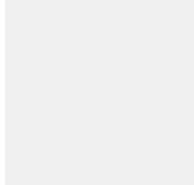
Seleccione una:

- a. El valor de 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. ✓

## Pregunta 11

Incorrecta

Puntúa como 1,00



Marcar pregunta

### Texto de la pregunta

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

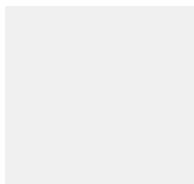
Seleccione una:

- a. Sí, siempre es así.
- b. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
- c. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir. 

## Pregunta 12

Correcta

Puntúa como 1,00



Marcar pregunta

### Texto de la pregunta

¿Para qué sirven las cotas pesimistas en *ramificación y poda*?

Seleccione una:

- a. Para descartar nodos basándose en la preferencia por algún otro nodo ya completado.
- b. Para tener la certeza de que la cota optimista está bien calculada.
- c. Para descartar nodos basándose en el beneficio esperado. 

# Parcial 3

En los algoritmos de ramificación y poda ...

Seleccione una:

- 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. El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida.

Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila:

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor óptimo de la mochila continua correspondiente.
- c. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.

Si para resolver un mismo problema usamos un algoritmo de ramificación y poda y lo modificamos minimamente para convertirlo en un algoritmo de vuelta atrás, ¿qué cambiamos realmente?

Seleccione una:

- a. Cambiamos la función que damos a la cota pesimista.
- b. Provocamos que las cotas optimistas pierdan eficacia.
- c. Será necesario comprobar si las soluciones son factibles o no puesto que ramificación y poda sólo genera nodos factibles.

Di cuál de estas tres soluciones a problemas de optimización no comporta, en el peor caso, tener que considerar  $O(72!)$  posibilidades.

Seleccione una:

- a. La solución de ramificación y poda al problema de la asignación de  $72$  tareas a  $72$  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. La solución al problema de buscar un árbol que cubre todos los vértices de un grafo de  $72$  vértices de forma que el coste es mínimo (*minimum spanning tree*).
- c. 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  $72$  vértices donde cada arista tiene un coste asignado.

Al resolver el problema del viajante de comercio mediante vuelta atrás y asumiendo un grafo de  $72$  vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Seleccione una:

- a. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las  $72$  aristas más cortas.
- c. Se multiplica  $72$  por la distancia de la arista más corta que nos queda por considerar.

Cuando resolvemos un problema mediante un esquema de ramificación y poda ...

Seleccione una:

- a. .... los valores entre los cuales se elige en cada una de las decisiones pueden formar un conjunto infinito.
- b. .... las decisiones sólo pueden ser binarias.
- c. .... los valores entre los cuales se elige en cada una de las decisiones tienen que formar un conjunto finito.

En los algoritmos de ramificación y poda, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
- b. No, nunca es así.
- c. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.

La complejidad en el mejor de los casos de un algoritmo de vuelta atrás ...

Seleccione una:

- a. .... es siempre exponencial con el número de decisiones a tomar.
- b. .... suele ser polinómica con el número de alternativas por cada decisión.
- c. .... puede ser polinómica con el número de decisiones a tomar.

En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento ...

Seleccione una:

- a. .... pueden eliminar soluciones parciales que son factibles.
- b. .... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- c. Las dos anteriores son verdaderas.

En la estrategia de ramificación y poda ...

Seleccione una:

- a. .... cada nodo tiene su propia cota optimista, la cota pesimista sin embargo, es común para todos los nodos.
- b. .... cada nodo tiene su propia cota pesimista, la cota optimista sin embargo, es común para todos los nodos.
- c. .... cada nodo tiene su propia cota pesimista y también su propia cota optimista.

Cuando se resuelve usando un algoritmo de ramificación y poda un problema de  $72$  decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a.  $O(n!)$
- b.  $O(n^2)$
- c.  $O(2^n)$

Se desea encontrar el camino mas corto entre dos ciudades

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

Seleccione una:

- a. Sería una cota optimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.
- b. Sería una cota pesimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.
- c. Ninguna de las otras dos opciones.

## TERCER PARCIAL

Texto de la pregunta

Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila:

Seleccione una:

- a. El valor óptimo 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. X

Texto de la pregunta

Di cuál de estas tres soluciones a problemas de optimización no comporta, en el peor caso, tener que considerar  $O(n!)$  posibilidades.

Seleccione una:

- a. La solución de *vuelta atrás* al problema del viajante de comercio (*travelling salesman problem*), o sea, el de encontrar un ciclo hamiltoniano de coste mínimo en un grafo conexo de  $n$  vértices donde cada arista tiene un coste asignado.
- b. La solución de *ramificación y poda* al problema de la asignación de  $m$  tareas a  $n$  trabajadores de forma que cada trabajador hace exactamente una tarea y cada tarea es asignada a un trabajador exactamente, de forma que la suma de los costes de las tareas es mínimo. X
- c. La solución al problema de buscar un árbol que cubre todos los vértices de un grafo de  $n$  vértices de forma que el coste es mínimo (*minimum spanning tree*).

Texto de la pregunta

¿Para qué sirven las cotas pesimistas en *ramificación y poda*?

Seleccione una:

- a. Para tener la certeza de que la cota optimista está bien calculada.
- b. Para descartar nodos basándose en la preferencia por algún otro nodo ya completado.
- c. Para descartar nodos basándose en el beneficio esperado. ✓

Texto de la pregunta

En los algoritmos de *ramificación y poda* ...

Seleccione una:

- a. El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida.
- b. Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.
- c. Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima. ✓

Texto de la pregunta

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo,  $\$-1\$$ ) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. 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.

Seleccione una:

- a. El nuevo algoritmo solo será más rápido para algunas instancias del problema. ✓
- b. Esta estrategia no asegura que se obtenga el camino mas corto.
- c. El nuevo algoritmo siempre sea más rápido.

Texto de la pregunta

Si para resolver un mismo problema usamos un algoritmo de *ramificación y poda* y lo modificamos mínimamente para convertirlo en un algoritmo de *vuelta atrás*, ¿qué cambiamos realmente?

Seleccione una:

- a. Cambiamos la función que damos a la cota pesimista.
- b. Provocamos que las cotas optimistas pierdan eficacia.
- c. Sería necesario comprobar si las soluciones son factibles o no puesto que *ramificación y poda* sólo genera nodos factibles. ✗

Texto de la pregunta

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
- b. Sí, siempre es así.
- c. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir. ✓

Texto de la pregunta

Al resolver el problema del viajante de comercio mediante *vuelta atrás*, ¿cuál de estas cotas optimistas se espera que padezca el árbol de búsqueda?

Seleccione una:

- 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 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.
- c. Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.

Texto de la pregunta

Cuando se resuelve usando un algoritmo de ramificación y poda un problema de  $\text{f!}$  decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a.  $O(n!)$  X
- b.  $O(n^2)$
- c.  $O(2^n)$

Texto de la pregunta

En ausencia de cotas optimistas y pesimistas, la estrategia de *vuelta atrás* ...

Seleccione una:

- a. ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.
- b. ... debe recorrer siempre todo el árbol. X
- c. ... no se puede usar para resolver problemas de optimización.

Texto de la pregunta

Cuando resolvemos un problema mediante un esquema de ramificación y poda ...

Seleccione una:

- a. ... las decisiones sólo pueden ser binarias.
- b. ... los valores entre los cuales se elige en cada una de las decisiones pueden formar un conjunto infinito. X
- c. ... los valores entre los cuales se elige en cada una de las decisiones tienen que formar un conjunto finito.

Texto de la pregunta

La complejidad en el peor de los casos de un algoritmo de *ramificación y poda* ...

Seleccione una:

- a. ... puede ser exponencial con el número de alternativas por cada decisión.
- b. ... puede ser polinómica con el número de decisiones a tomar. X
- c. ... es exponencial con el número de decisiones a tomar.

|                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Pregunta 1</b><br>Incorrecta<br>Puntúa como 1.00<br><input type="checkbox"/> Marcar pregunta     | <p>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 para como cota para limitar la búsqueda en un algoritmo de <i>vuelta atrás</i>.</p> <p>¿Qué tipo de cota sería?</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. Una cota optimista.</li> <li><input checked="" type="radio"/> b. Una cota pesimista. <span style="color: red;">X</span></li> <li><input type="radio"/> c. No se trataría de ninguna poda puesto que es posible que esa heurística no encuentre una solución factible.</li> </ul> |
| <b>Pregunta 2</b><br>Incorrecta<br>Puntúa como 1.00<br><input type="checkbox"/> Marcar pregunta     | <p>La estrategia de ramificación y poda necesita cotas pesimistas ...</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. .... sólo si se usa para resolver problemas de optimización.</li> <li><input type="radio"/> b. .... para determinar si una solución es factible.</li> <li><input checked="" type="radio"/> c. .... para decidir el orden de visita de los nodos del árbol de soluciones. <span style="color: red;">X</span></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Pregunta 3</b><br>Correcta<br>Puntúa como 1.00<br><input type="checkbox"/> Marcar pregunta       | <p>En los algoritmos de <i>ramificación y poda</i> ...</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. Una cota pesimista es necesariamente un valor inalcanzable, de no ser así no está garantizado que no se eliminen nodos factibles.</li> <li><input checked="" type="radio"/> b. Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima. <span style="color: green;">✓</span></li> <li><input type="radio"/> c. Una cota pesimista es el beneficio esperado de cualquier nodo factible.</li> </ul>                                                                                                                                                                                                                                                                                                                                                    |
| <b>Pregunta 4</b><br>Correcta<br>Puntúa como 1.00<br><input type="checkbox"/> Marcar pregunta       | <p>Al resolver el problema del viajante de comercio mediante <i>vuelta atrás</i>, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las <math>k_C</math> aristas más cortas, donde <math>k_C</math> es el número de saltos que nos quedan por dar. <span style="color: green;">✓</span></li> <li><input type="radio"/> b. Se multiplica <math>k_C</math> por la distancia de la arista más corta que nos queda por considerar, donde <math>k_C</math> es el número de saltos que nos quedan por dar.</li> <li><input type="radio"/> c. Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.</li> </ul>                                                                              |
| <b>Pregunta 5</b><br>Sin contestar<br>Puntúa como 1.00<br><input type="checkbox"/> Marcar pregunta  | <p>Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila:</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.</li> <li><input type="radio"/> b. El valor óptimo de la mochila continua correspondiente.</li> <li><input type="radio"/> c. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.</li> </ul>                                                                                                                                                                                                                                                                                                                                                      |
| <b>Pregunta 6</b><br>Correcta<br>Puntúa como 1.00<br><input type="checkbox"/> Marcar pregunta       | <p>En el esquema de <i>vuelta atrás</i>, los mecanismos de poda basados en la mejor solución hasta el momento ...</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. .... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.</li> <li><input checked="" type="radio"/> b. .... pueden eliminar soluciones parciales que son factibles. <span style="color: green;">✓</span></li> <li><input type="radio"/> c. Las dos anteriores son verdaderas.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Pregunta 7</b><br>Incorrecta<br>Puntúa como 1.00<br><input type="checkbox"/> Marcar pregunta     | <p>Tratándose de un problema de optimización, en la lista de nodos vivos de <i>ramificación y poda</i> ...</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. .... puede haber nodos que no son prometedores.</li> <li><input type="radio"/> b. Las otras dos opciones son ciertas.</li> <li><input checked="" type="radio"/> c. .... sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento. <span style="color: red;">X</span></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>Pregunta 8</b><br>Correcta<br>Puntúa como 1.00<br><input type="checkbox"/> Marcar pregunta       | <p>Cuando se resuelve usando un algoritmo de <i>vuelta atrás</i> un problema de <math>\gamma_2</math> decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. <math>O(n!)</math></li> <li><input checked="" type="radio"/> b. <math>O(2^{\gamma_2})</math> <span style="color: green;">✓</span></li> <li><input type="radio"/> c. <math>O(n^2)</math></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Pregunta 9</b><br>Correcta<br>Puntúa como 1.00<br><input type="checkbox"/> Marcar pregunta       | <p>En los algoritmos de <i>ramificación y poda</i>, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.</li> <li><input type="radio"/> b. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir. <span style="color: green;">✓</span></li> <li><input type="radio"/> c. Si, siempre es así.</li> </ul>                                                                                                                                                                                                                                                                                                                   |
| <b>Pregunta 10</b><br>Sin contestar<br>Puntúa como 1.00<br><input type="checkbox"/> Marcar pregunta | <p>El problema de cortar un tubo de longitud <math>\gamma_2</math> en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. .... no se puede resolver usando un algoritmo de <i>vuelta atrás</i>.</li> <li><input type="radio"/> b. .... se debe resolver mediante un algoritmo de <i>vuelta atrás</i>, dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo.</li> <li><input type="radio"/> c. .... se puede resolver mediante un algoritmo de <i>vuelta atrás</i> pero existe una solución asintóticamente mucho más eficiente.</li> </ul>                                                                                                                                                                                                                    |

**Pregunta 11**

Correcta

Puntúa como 1,00

Marcar pregunta

La complejidad en el mejor de los casos de un algoritmo de *ramificación y poda*...

Seleccione una:

- a. ... es siempre exponencial con el número de decisiones a tomar.
- b. ... suele ser polinómica con el número de alternativas por cada decisión
- c. ... puede ser polinómica con el número de decisiones a tomar. ✓

**Pregunta 12**

Incorrecta

Puntúa como 1,00

Marcar pregunta

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?

Seleccione una:

- a. Aprovechamos mejor las cotas optimistas.
- b. La comprobación de las soluciones factibles: en *ramificación y poda* no es necesario puesto que sólo genera nodos factibles. ✗
- c. Cambiamos la función que damos a la cota pesimista.

## Preguntas ADA tercer parcial

**1.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 para como cota para limitar la búsqueda en un algoritmo de *vuelta atrás*.

una cota optimista.

**2.Se desea obtener todas las permutaciones de una lista compuesta por N elementos, ¿Que esquema es el más adecuado?**

Seleccione una:

*Vuelta atrás*, para este problema no hay un esquema más eficiente.

**3.Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:**

El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

**4.En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)**

En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.

**5.En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista?**

En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.

**6.La complejidad en el peor de los casos de un algoritmo de vuelta atrás ...**

... es exponencial con el número de decisiones a tomar.

**7.La complejidad en el peor de los casos de un algoritmo de *ramificación y poda* ...**

es exponencial con el número de decisiones a tomar.

**8.La estrategia de vuelta atrás es aplicable a problemas de selección y optimización en los que:**

El espacio de soluciones es un conjunto finito.

**9. ¿Para qué sirven las cotas pesimistas en ramificación y poda? Seleccione una:**

Para descartar nodos basándose en el beneficio esperado.

**10. 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 en el caso peor es la mejor que nos podemos encontrar?**

$O(2^n)$

**10. Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca el árbol de búsqueda?**

Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las aristas más cortas, donde  $k$  es el número de saltos que nos quedan por dar.

**11. La ventaja de la estrategia ramificación y poda frente a vuelta atrás es que la primera genera las soluciones posibles al problema mediante ...**

Las otras dos opciones son verdaderas.

**12. En los algoritmos de ramificación y poda ...**

Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.

**13. La estrategia de ramificación y poda genera las soluciones posibles al problema mediante ...**

... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.

**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?**

El orden de exploración de las soluciones.

**15. El problema de cortar un tubo de longitud  $n$  en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...**

---

**16. Cuando resolvemos un problema mediante un esquema de ramificación y poda ...**

... los valores entre los cuales se elige en cada una de las decisiones tienen que formar un conjunto finito.

**17. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás ...**

... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.

**18.Al resolver el problema del viajante de comercio mediante *vuelta atrás* y asumiendo un grafo de N vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?**

Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.

**19.Tratándose de un problema de optimización, en la lista de nodos vivos de *ramificación y poda* ...**

Las otras dos opciones son ciertas.

**20.Dado un problema de optimización cualquiera, ¿la estrategia de *vuelta atrás* garantiza la solución óptima?**

Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.

**21. Se desea encontrar el camino mas corto entre dos ciudades.Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo *voraz* basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.**

**¿Qué tipo de cota sería?**

Sería una **cota pesimista** siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.

**22.La complejidad en el mejor de los casos de un algoritmo de *ramificación y poda*...**

... puede ser polinómica con el número de decisiones a tomar.

**23.El uso de funciones de cota en ramificación y poda...**

puede reducir el numero de instancias del problema que pertenecen al caso peor.

**24.En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento...**

...pueden eliminar soluciones parciales que son factibles.

**25.Di cuál de estas tres soluciones a problemas de optimización no comporta, en el peor caso, tener que considerar  $O(n!)$  posibilidades.**

-----

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 en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a.  $O(2^n)$  ✓
- b.  $O(n!)$
- c.  $O(n^2)$

Al resolver el problema del viajante de comercio mediante *vuelta atrás* y asumiendo un grafo de  $n$  vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Seleccione una:

- a. Se multiplica  $n$  por la distancia de la arista más corta que nos queda por considerar.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las  $n$  aristas más cortas.
- c. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido. ✓

Se desea obtener todas las permutaciones de una lista compuesta por  $n$  elementos. ¿Qué esquema es el más adecuado?

Seleccione una:

- a. *Ramificación y poda*, puesto que con buenas funciones de cota es más eficiente para este problema que *vuelta atrás*.
- b. *Divide y vencerás*, puesto que la división en sublistas se podría hacer en tiempo constante.
- c. *Vuelta atrás*, para este problema no hay un esquema más eficiente. ✓

La complejidad en el mejor de los casos de un algoritmo de *ramificación y poda* ...

Seleccione una:

- a. ... es siempre exponencial con el número de decisiones a tomar.
- b. ... suele ser polinómica con el número de alternativas por cada decisión
- c. ... puede ser polinómica con el número de decisiones a tomar. ✓

La complejidad en el peor de los casos de un algoritmo de *ramificación y poda* ...

Seleccione una:

- a. ... puede ser exponencial con el número de alternativas por cada decisión.
- b. ... puede ser polinómica con el número de decisiones a tomar.
- c. ... es exponencial con el número de decisiones a tomar. ✓

La estrategia de *ramificación y poda* genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones. ✓
- b. ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- c. ... un recorrido en anchura del árbol que representa el espacio de soluciones.

¿Para qué sirven las cotas pesimistas en *ramificación y poda*?

Seleccione una:

- a. Para tener la certeza de que la cota optimista está bien calculada.
- b. Para descartar nodos basándose en la preferencia por algún otro nodo ya completado.
- c. Para descartar nodos basándose en el beneficio esperado. ✓

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- 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, nunca es así.

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
- b. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir. ✓
- c. Sí, siempre es así.

En los algoritmos de *ramificación y poda* ...

Seleccione una:

- a. El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida.
- 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 optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.

La ventaja de la estrategia *ramificación y poda* frente a *vuelta atrás* es que la primera genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.
- b. Las otras dos opciones son verdaderas. ✓
- c. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.

¿Cuál es la diferencia principal entre una solución de *vuelta atrás* y una solución de *ramificación y poda* para el problema de la mochila?

Seleccione una:

- a. El orden de exploración de las soluciones. ✓
- b. El coste asintótico en el caso peor.
- c. El hecho que la solución de *ramificación y poda* puede empezar con una solución subóptima voraz y la de *vuelta atrás* no.

Tratándose de un problema de optimización, en la lista de nodos vivos de *ramificación y poda* ...

Seleccione una:

- a. ... sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento.
- b. ... puede haber nodos que no son prometedores.
- c. Las otras dos opciones son ciertas. ✓

Cuando resolvemos un problema mediante un esquema de *ramificación y poda* ...

Seleccione una:

- a. ... los valores entre los cuales se elige en cada una de las decisiones tienen que formar un conjunto finito. ✓
- b. ... las decisiones sólo pueden ser binarias.
- c. ... los valores entre los cuales se elige en cada una de las decisiones pueden formar un conjunto infinito.

La estrategia de ramificación y poda necesita cotas pesimistas ...

Seleccione una:

- a. ... para decidir el orden de visita de los nodos del árbol de soluciones.
- b. ... sólo si se usa para resolver problemas de optimización. ✓
- c. ... para determinar si una solución es factible.

El uso de funciones de cota en ramificación y poda ...

Seleccione una:

- a. ... transforma en polinómicas complejidades que antes eran exponenciales.
- b. ... puede reducir el número de instancias del problema que pertenecen al caso peor. ✓
- c. ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.

En la estrategia de *ramificación y poda* ...

Seleccione una:

- a. ... cada nodo tiene su propia cota pesimista y también su propia cota optimista. ✓
- b. ... cada nodo tiene su propia cota pesimista, la cota optimista sin embargo, es común para todos los nodos.
- c. ... cada nodo tiene su propia cota optimista, la cota pesimista sin embargo, es común para todos los nodos.

Si para resolver un mismo problema usamos un algoritmo de *vuelta atrás* y lo modificamos mínimamente para convertirlo en un algoritmo de *ramificación y poda*, ¿qué cambiamos realmente?

Seleccione una:

- a. La comprobación de las soluciones factibles: en *ramificación y poda* no es necesario puesto que sólo genera nodos factibles. ✗
- b. Cambiamos la función que damos a la cota pesimista.
- c. Aprovechamos mejor las cotas optimistas.

Si para resolver un mismo problema usamos un algoritmo de *ramificación y poda* y lo modificamos mínimamente para convertirlo en un algoritmo de *vuelta atrás*, ¿qué cambiamos realmente?

Seleccione una:

- a. Cambiamos la función que damos a la cota pesimista.
- b. Provocamos que las cotas optimistas pierdan eficacia.
- c. Será necesario comprobar si las soluciones son factibles o no puesto que *ramificación y poda* sólo genera nodos factibles.

En el esquema de *vuelta atrás*, los mecanismos de poda basados en la mejor solución hasta el momento ...

Seleccione una:

- a. Las dos anteriores son verdaderas.
- b. ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- c. ... pueden eliminar soluciones parciales que son factibles. ✓

En ausencia de cotas optimistas y pesimistas, la estrategia de *vuelta atrás* ...

Seleccione una:

- a. ... no se puede usar para resolver problemas de optimización.
- b. ... debe recorrer siempre todo el árbol.
- c. ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles. ✓

La estrategia de *vuelta atrás* es aplicable a problemas de selección y optimización en los que:

Seleccione una:

- a. El espacio de soluciones puede ser tanto finito como infinito pero en este último caso debe ser al menos numerable.
- b. El espacio de soluciones es un conjunto infinito.
- c. El espacio de soluciones es un conjunto finito. ✓

Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila:

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- c. El valor óptimo de la mochila continua correspondiente.

Decid cuál de estas tres no sirve como cota optimista para obtener el valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- b. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos. ✓
- c. El valor de la mochila continua correspondiente.

Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor de la mochila continua correspondiente.
- c. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

Seleccione una:

- a. podemos aplicar el esquema *vuelta atrás* siempre que se trate de un conjunto infinito numerable.
- b. es probable que a través de *programación dinámica* se obtenga un algoritmo eficaz que lo solucione.
- c. una estrategia voraz puede ser la única alternativa. ✓

Dado un problema de optimización cualquiera, ¿la estrategia de *vuelta atrás* garantiza la solución óptima?

Seleccione una:

- a. Es condición necesaria que el dominio de las decisiones sea discreto o discretizable y que el número de decisiones a tomar esté acotado.
- b. Sí, puesto que ese método analiza todas las posibilidades.
- c. 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.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

Seleccione una:

- a. Sería una **cota pesimista** siempre que se tenga la certeza de que esa aproximación encuentra una solución factible. ✓
- b. Ninguna de las otras dos opciones.
- c. Sería una **cota optimista** siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. Como también se conocen las coordenadas geográficas de cada ciudad se quiere usar la distancia geográfica (en línea recta) entre cada par de ciudades para como cota para limitar la búsqueda en un algoritmo de *vuelta atrás*.

¿Qué tipo de cota sería?

Seleccione una:

- a. No se trataría de ninguna poda puesto que esa heurística no encuentre una solución factible.
- b. Una cota pesimista.
- c. Una cota optimista. ✓

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, \$-1\$) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. 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.

Seleccione una:

- a. El nuevo algoritmo solo será más rápido para algunas instancias del problema. ✓
- b. Esta estrategia no asegura que se obtenga el camino mas corto.
- c. El nuevo algoritmo siempre sea más rápido.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino según su distancia geográfica.

Este algoritmo voraz, ¿serviría como cota pesimista?

Seleccione una:

- a. No, ya que no asegura que se encuentre una solución factible. ✓
- b. Sí, puesto que la distancia geográfica asegura que otra solución mejor no es posible.
- c. No, ya que en algunos casos puede dar distancias menores que la óptima.

El problema de cortar un tubo de longitud  $\gamma$  en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...

Seleccione una:

- a. ... no se puede resolver usando un algoritmo de *vuelta atrás*.
- b. ... se puede resolver mediante un algoritmo de *vuelta atrás* pero existe una solución asintóticamente mucho más eficiente. ✓
- c. ... se debe resolver mediante un algoritmo de *vuelta atrás*, dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo.

Al resolver el problema del viajante de comercio mediante *vuelta atrás* y asumiendo un grafo de  $\gamma$  vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Seleccione una:

- a. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las  $\gamma$  aristas más cortas.
- c. Se multiplica  $\gamma$  por la distancia de la arista más corta que nos queda por considerar.

Al resolver el problema del viajante de comercio mediante *vuelta atrás*, ¿cuál de estas cotas optimistas se espera que padezca el árbol de búsqueda?

Seleccione una:

- 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. ✓

Di cuál de estas tres soluciones a problemas de optimización no comporta, en el peor caso, tener que considerar  $O(n!)$  posibilidades.

Seleccione una:

- a. La solución de *vuelta atrás* al problema del viajante de comercio (*travelling salesman problem*), o sea, el de encontrar un ciclo hamiltoniano de coste mínimo en un grafo conexo de  $n$  vértices donde cada arista tiene un coste asignado. X
- b. La solución de *ramificación y poda* al problema de la asignación de  $n$  tareas a  $n$  trabajadores de forma que cada trabajador hace exactamente una tarea y cada tarea es asignada a un trabajador exactamente, de forma que la suma de los costes de las tareas es mínimo. X
- c. La solución al problema de buscar un árbol que cubre todos los vértices de un grafo de  $n$  vértices de forma que el coste es mínimo (*minimum spanning tree*).

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 en el caso peor es la mejor que nos podemos encontrar?

Seleccione una:

- a.  $O(2^n)$  ✓
- b.  $O(n!)$
- c.  $O(n^2)$

Al resolver el problema del viajante de comercio mediante *vuelta atrás* y asumiendo un grafo de  $n$  vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Seleccione una:

- a. Se multiplica  $n$  por la distancia de la arista más corta que nos queda por considerar.
- b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las  $n$  aristas más cortas.
- c. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido. ✓

Se desea obtener todas las permutaciones de una lista compuesta por  $n$  elementos. ¿Qué esquema es el más adecuado?

Seleccione una:

- a. *Ramificación y poda*, puesto que con buenas funciones de cota es más eficiente para este problema que *vuelta atrás*.
- b. *Divide y vencerás*, puesto que la división en sublistas se podría hacer en tiempo constante.
- c. *Vuelta atrás*, para este problema no hay un esquema más eficiente. ✓

La complejidad en el mejor de los casos de un algoritmo de *ramificación y poda*...

Seleccione una:

- a. ... es siempre exponencial con el número de decisiones a tomar.
- b. ... suele ser polinómica con el número de alternativas por cada decisión
- c. ... puede ser polinómica con el número de decisiones a tomar. ✓

La complejidad en el peor de los casos de un algoritmo de *ramificación y poda* ...

Seleccione una:

- a. ... puede ser exponencial con el número de alternativas por cada decisión.
- b. ... puede ser polinómica con el número de decisiones a tomar.
- c. ... es exponencial con el número de decisiones a tomar. ✓

La estrategia de *ramificación y poda* genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones. ✓
- b. ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- c. ... un recorrido en anchura del árbol que representa el espacio de soluciones.

¿Para qué sirven las cotas pesimistas en *ramificación y poda*?

Seleccione una:

- a. Para tener la certeza de que la cota optimista está bien calculada.
- b. Para descartar nodos basándose en la preferencia por algún otro nodo ya completado.
- c. Para descartar nodos basándose en el beneficio esperado. ✓

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- 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, nunca es así.

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.
- b. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir. ✓
- c. Sí, siempre es así.

En los algoritmos de *ramificación y poda* ...

Seleccione una:

- a. El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida.
- 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 optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.

La ventaja de la estrategia *ramificación y poda* frente a *vuelta atrás* es que la primera genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido guiado por una cola de prioridad de donde se extraen primero los nodos que representan los subárboles más prometedores del espacio de soluciones.
- b. Las otras dos opciones son verdaderas. ✓
- c. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.

¿Cuál es la diferencia principal entre una solución de *vuelta atrás* y una solución de *ramificación y poda* para el problema de la mochila?

Seleccione una:

- a. El orden de exploración de las soluciones. ✓
- b. El coste asintótico en el caso peor.
- c. El hecho que la solución de *ramificación y poda* puede empezar con una solución subóptima voraz y la de *vuelta atrás* no.

Tratándose de un problema de optimización, en la lista de nodos vivos de *ramificación y poda* ...

Seleccione una:

- a. ... sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento.
- b. ... puede haber nodos que no son prometedores.
- c. Las otras dos opciones son ciertas. ✓

Cuando resolvemos un problema mediante un esquema de *ramificación y poda* ...

Seleccione una:

- a. ... los valores entre los cuales se elige en cada una de las decisiones tienen que formar un conjunto finito. ✓
- b. ... las decisiones sólo pueden ser binarias.
- c. ... los valores entre los cuales se elige en cada una de las decisiones pueden formar un conjunto infinito.

La estrategia de ramificación y poda necesita cotas pesimistas ...

Seleccione una:

- a. ... para decidir el orden de visita de los nodos del árbol de soluciones.
- b. ... sólo si se usa para resolver problemas de optimización. ✓
- c. ... para determinar si una solución es factible.

El uso de funciones de cota en ramificación y poda ...

Seleccione una:

- a. ... transforma en polinómicas complejidades que antes eran exponenciales.
- b. ... puede reducir el número de instancias del problema que pertenecen al caso peor. ✓
- c. ... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.

En el esquema de *vuelta atrás*, los mecanismos de poda basados en la mejor solución hasta el momento ...

Seleccione una:

- a. Las dos anteriores son verdaderas.
- b. ... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- c. ... pueden eliminar soluciones parciales que son factibles. ✓

En ausencia de cotas optimistas y pesimistas, la estrategia de *vuelta atrás* ...

Seleccione una:

- a. ... no se puede usar para resolver problemas de optimización.
- b. ... debe recorrer siempre todo el árbol.
- c. ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles. ✓

La estrategia de *vuelta atrás* es aplicable a problemas de selección y optimización en los que:

Seleccione una:

- a. El espacio de soluciones puede ser tanto finito como infinito pero en este último caso debe ser al menos numerable.
- b. El espacio de soluciones es un conjunto infinito.
- c. El espacio de soluciones es un conjunto finito. ✓

Decid cuál de estas tres no sirve como cota optimista para obtener el valor óptimo de la mochila discreta:

Seleccione una:

- a. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- b. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos. ✓
- c. El valor de la mochila continua correspondiente.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

Seleccione una:

- a. Sería una **cota pesimista** siempre que se tenga la certeza de que esa aproximación encuentra una solución factible. ✓
- b. Ninguna de las otras dos opciones.
- c. Sería una **cota optimista** siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. Como también se conocen las coordenadas geográficas de cada ciudad se quiere usar la distancia geográfica (en línea recta) entre cada par de ciudades para como cota para limitar la búsqueda en un algoritmo de *vuelta atrás*.

¿Qué tipo de cota seria?

Seleccione una:

- a. No se trataría de ninguna poda puesto que esa heurística no encuentre una solución factible.
- b. Una cota pesimista.
- c. Una cota optimista. ✓

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. 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.

Seleccione una:

- a. El nuevo algoritmo solo será más rápido para algunas instancias del problema. ✓
- b. Esta estrategia no asegura que se obtenga el camino mas corto.
- c. El nuevo algoritmo siempre sea más rápido.

# 2012-13\_ANALISIS Y DISEÑO DE ALGORITMOS\_34018

Página Principal ► Mis cursos ► ADA\_34018 ► Tercer examen parcial de ADA ► Tercer parcial

**Pregunta 1** La complejidad en el mejor de los casos de un algoritmo de *ramificación y poda*...

Incorrecta

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- a. .... puede ser polinómica con el número de decisiones a tomar.
- b. .... suele ser polinómica con el número de alternativas por cada decisión
- c. .... es siempre exponencial con el número de decisiones a tomar.

**Pregunta 2**

En los algoritmos de *ramificación y poda* ...

Correcta

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- 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 beneficio esperado de cualquier nodo factible que no es el óptimo.

**Pregunta 3**

¿Cuál es la diferencia principal entre una solución de *vuelta atrás* y una solución de *ramificación y poda* para el problema de la mochila?

Correcta

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- a. El orden de exploración de las soluciones.
- b. El coste asintótico en el caso peor.
- c. El hecho que la solución de *ramificación y poda* puede empezar con una solución subóptima voraz y la de *vuelta atrás* no.

**Pregunta 4**

Al resolver el problema del viajante de comercio mediante *vuelta atrás* y asumiendo un grafo de  $n$  vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?

Correcta

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- a. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.
- b. Se multiplica  $n$  por la distancia de la arista más corta que nos queda por considerar.
- c. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las  $n$  aristas más cortas.

**Pregunta 5**

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 en el caso peor es la mejor que nos podemos encontrar?

Correcta

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- a.  $O(n^2)$
- b.  $O(n!)$
- c.  $O(2^n)$

**Pregunta 6**

Decid cuál de estas tres no sirve como cota optimista para obtener el valor óptimo de la mochila discreta:

Incorrecta

Puntúa como 1,00

Marcar pregunta

Seleccione una:

- a. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- b. El valor de la mochila continua correspondiente.
- c. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el

valor específico de los objetos.

**Pregunta 7**

Sin contestar

Puntúa como 1,00

 Marcar pregunta

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 para como cota para limitar la búsqueda en un algoritmo de *vuelta atrás*.

¿Qué tipo de cota sería?

Seleccione una:

- a. No se trataría de ninguna poda puesto que es posible que esa heurística no encuentre una solución factible.
- b. Una cota optimista.
- c. Una cota pesimista.

**Pregunta 8**

Correcta

Puntúa como 1,00

 Marcar pregunta

Se desea obtener todas las permutaciones de una lista compuesta por  $n$  elementos. ¿Qué esquema es el más adecuado?

Seleccione una:

- a. *Vuelta atrás*, para este problema no hay un esquema más eficiente. ✓
- 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 para este problema que *vuelta atrás*.

**Pregunta 9**

Correcta

Puntúa como 1,00

 Marcar pregunta

El uso de funciones de cota en ramificación y poda ...

Seleccione una:

- a. .... transforma en polinómicas complejidades que antes eran exponenciales.
- b. .... puede reducir el número de instancias del problema que pertenecen al caso peor. ✓
- c. .... garantiza que el algoritmo va a ser más eficiente ante cualquier instancia del problema.

**Pregunta 10**

Sin contestar

Puntúa como 1,00

 Marcar pregunta

Tratándose de un problema de optimización, en la lista de nodos vivos de *ramificación y poda* ...

Seleccione una:

- a. .... puede haber nodos que no son prometedores.
- b. .... sólo se introducen nodos prometedores, es decir, nodos que pueden mejorar la mejor solución que se tiene en ese momento.
- c. Las otras dos opciones son ciertas.

**Pregunta 11**

Incorrecta

Puntúa como 1,00

 Marcar pregunta

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. Sí, siempre es así.
- b. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.
- c. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir. X

**Pregunta 12**

Sin contestar

Puntúa como 1,00

 Marcar pregunta

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

Seleccione una:

- a. una estrategia voraz puede ser la única alternativa.
- b. podremos aplicar el esquema *vuelta atrás* siempre que se trate de un conjunto infinito numerable.
- c. es probable que a través de *programación dinámica* se obtenga un algoritmo eficaz que lo

solucion.

|                                                                                                     |                                                                                                                                                                                                                                                              |
|-----------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Pregunta 1</b>                                                                                   | Si para resolver un mismo problema usamos un algoritmo de <i>vuelta atrás</i> y lo modificamos mínimamente para convertirlo en un algoritmo de <i>ramificación y poda</i> , ¿qué cambiamos realmente?                                                        |
| Incorrecta                                                                                          |                                                                                                                                                                                                                                                              |
| Puntúa como 1,00                                                                                    |                                                                                                                                                                                                                                                              |
|  Marcar pregunta   | Seleccione una:                                                                                                                                                                                                                                              |
|                                                                                                     | <input checked="" type="radio"/> a. La comprobación de las soluciones factibles: en <i>ramificación y poda</i> no es necesario puesto que sólo genera nodos factibles. <span style="color:red">X</span>                                                      |
|                                                                                                     | <input type="radio"/> b. Cambiamos la función que damos a la cota pesimista.                                                                                                                                                                                 |
|                                                                                                     | <input type="radio"/> c. Aprovechamos mejor las cotas optimistas.                                                                                                                                                                                            |
| <b>Pregunta 2</b>                                                                                   | La estrategia de <i>ramificación y poda</i> genera las soluciones posibles al problema mediante ...                                                                                                                                                          |
| Correcta                                                                                            |                                                                                                                                                                                                                                                              |
| Puntúa como 1,00                                                                                    |                                                                                                                                                                                                                                                              |
|  Marcar pregunta   | Seleccione una:                                                                                                                                                                                                                                              |
|                                                                                                     | <input type="radio"/> a. ... un recorrido en profundidad del árbol que representa el espacio de soluciones.                                                                                                                                                  |
|                                                                                                     | <input checked="" type="radio"/> b. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones. <span style="color:green">✓</span>                                                                      |
|                                                                                                     | <input type="radio"/> c. ... un recorrido en anchura del árbol que representa el espacio de soluciones.                                                                                                                                                      |
| <b>Pregunta 3</b>                                                                                   | Cuando se resuelve usando un algoritmo de ramificación y poda un problema de $n$ decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar? |
| Correcta                                                                                            |                                                                                                                                                                                                                                                              |
| Puntúa como 1,00                                                                                    |                                                                                                                                                                                                                                                              |
|  Marcar pregunta   | Seleccione una:                                                                                                                                                                                                                                              |
|                                                                                                     | <input checked="" type="radio"/> a. $O(2^n)$ <span style="color:green">✓</span>                                                                                                                                                                              |
|                                                                                                     | <input type="radio"/> b. $O(n^2)$                                                                                                                                                                                                                            |
|                                                                                                     | <input type="radio"/> c. $O(n!)$                                                                                                                                                                                                                             |
| <b>Pregunta 4</b>                                                                                   | Al resolver el problema del viajante de comercio mediante <i>vuelta atrás</i> , ¿cuál de estas cotas optimistas se espera que pade mejor el árbol de búsqueda?                                                                                               |
| Correcta                                                                                            |                                                                                                                                                                                                                                                              |
| Puntúa como 1,00                                                                                    |                                                                                                                                                                                                                                                              |
|  Marcar pregunta | Seleccione una:                                                                                                                                                                                                                                              |
|                                                                                                     | <input type="radio"/> a. Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.                                                                                                  |
|                                                                                                     | <input checked="" type="radio"/> b. 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. <span style="color:green">✓</span>              |
|                                                                                                     | <input type="radio"/> c. 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>Pregunta 5</b>                                                                                   | La estrategia de <i>vuelta atrás</i> es aplicable a problemas de selección y optimización en los que:                                                                                                                                                        |
| Correcta                                                                                            |                                                                                                                                                                                                                                                              |
| Puntúa como 1,00                                                                                    |                                                                                                                                                                                                                                                              |
|  Marcar pregunta | Seleccione una:                                                                                                                                                                                                                                              |
|                                                                                                     | <input type="radio"/> a. El espacio de soluciones es un conjunto infinito.                                                                                                                                                                                   |
|                                                                                                     | <input checked="" type="radio"/> b. El espacio de soluciones es un conjunto finito. <span style="color:green">✓</span>                                                                                                                                       |
|                                                                                                     | <input type="radio"/> c. El espacio de soluciones puede ser tanto finito como infinito pero en este último caso debe ser al menos numerable.                                                                                                                 |
| <b>Pregunta 6</b>                                                                                   | En la estrategia de <i>ramificación y poda</i> ...                                                                                                                                                                                                           |
| Correcta                                                                                            |                                                                                                                                                                                                                                                              |
| Puntúa como 1,00                                                                                    |                                                                                                                                                                                                                                                              |
|  Marcar pregunta | Seleccione una:                                                                                                                                                                                                                                              |
|                                                                                                     | <input checked="" type="radio"/> a. ... cada nodo tiene su propia cota pesimista y también su propia cota optimista. <span style="color:green">✓</span>                                                                                                      |
|                                                                                                     | <input type="radio"/> b. ... cada nodo tiene su propia cota pesimista, la cota optimista sin embargo, es común para todos los nodos.                                                                                                                         |
|                                                                                                     | <input type="radio"/> c. ... cada nodo tiene su propia cota optimista, la cota pesimista sin embargo, es común para todos los nodos.                                                                                                                         |
| <b>Pregunta 7</b>                                                                                   | Decid cuál de estas tres es la cota pesimista más ajustada al valor óptimo de la mochila discreta:                                                                                                                                                           |
| Sin contestar                                                                                       |                                                                                                                                                                                                                                                              |
| Puntúa como 1,00                                                                                    |                                                                                                                                                                                                                                                              |

 Marcar pregunta

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.
- b. El valor de la mochila continua correspondiente.
- c. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.

**Pregunta 8**

Correcta

Puntúa como 1,00

 Marcar pregunta

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- a. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir. 
- b. Sí, siempre es así.
- c. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir.

**Pregunta 9**

Sin contestar

Puntúa como 1,00

 Marcar pregunta

La complejidad en el peor de los casos de un algoritmo de *ramificación y poda* ...

Seleccione una:

- a. ... es exponencial con el número de decisiones a tomar.
- b. ... puede ser exponencial con el número de alternativas por cada decisión.
- c. ... puede ser polinómica con el número de decisiones a tomar.

**Pregunta 10**

Sin contestar

Puntúa como 1,00

 Marcar pregunta

En los algoritmos de *ramificación y poda* ...

Seleccione una:

- a. Una cota pesimista es el beneficio esperado de cualquier nodo factible.
- b. Una cota pesimista es necesariamente un valor inalcanzable, de no ser así no está garantizado que no se eliminan nodos factibles.
- c. Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima.

**Pregunta 11**

Correcta

Puntúa como 1,00

 Marcar pregunta

El problema de cortar un tubo de longitud  $n$  en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...

Seleccione una:

- a. ... no se puede resolver usando un algoritmo de *vuelta atrás*.
- b. ... se debe resolver mediante un algoritmo de *vuelta atrás*, dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo.
- c. ... se puede resolver mediante un algoritmo de *vuelta atrás* pero existe una solución asintóticamente mucho más eficiente. 

**Pregunta 12**

Incorrecta

Puntúa como 1,00

 Marcar pregunta

Se desea encontrar el camino mas corto entre dos ciudades.

Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de *vuelta atrás*, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino según su distancia geográfica.

Este algoritmo voraz, ¿serviría como cota pesimista?

Seleccione una:

- a. No, ya que en algunos casos puede dar distancias menores que la óptima. 
- b. No, ya que no asegura que se encuentre una solución factible.
- c. Sí, puesto que la distancia geográfica asegura que otra solución mejor no es posible.

## TERCER PARCIAL

|                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Pregunta 1</b><br/>Correcta<br/>Puntúa como 1.00<br/><input type="button" value="Marcar pregunta"/></p> | <p>La estrategia de <i>vuelta atrás</i> es aplicable a problemas de selección y optimización en los que:</p> <p>Seleccione una:</p> <p><input type="radio"/> a. El espacio de soluciones puede ser tanto finito como infinito pero en este último caso debe ser al menos numerable.</p> <p><input type="radio"/> b. El espacio de soluciones es un conjunto infinito.</p> <p><input checked="" type="radio"/> c. El espacio de soluciones es un conjunto finito. ✓</p>                                                            |
| <p><b>Pregunta 2</b><br/>Correcta<br/>Puntúa como 1.00<br/><input type="button" value="Marcar pregunta"/></p> | <p>La estrategia de <i>ramificación y poda</i> genera las soluciones posibles al problema mediante ...</p> <p>Seleccione una:</p> <p><input type="radio"/> a. ... un recorrido en profundidad del árbol que representa el espacio de soluciones.</p> <p><input type="radio"/> b. ... un recorrido en anchura del árbol que representa el espacio de soluciones.</p> <p><input checked="" type="radio"/> c. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones. ✓</p> |
| <p><b>Pregunta 3</b><br/>Correcta<br/>Puntúa como 1.00<br/><input type="button" value="Marcar pregunta"/></p> | <p>Decid cuál de estas tres no sirve como cota optimista para obtener el valor óptimo de la mochila discreta:</p> <p>Seleccione una:</p> <p><input type="radio"/> a. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.</p> <p><input checked="" type="radio"/> b. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos. ✓</p> <p><input type="radio"/> c. El valor de la mochila continua correspondiente.</p>    |

|                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Pregunta 4</b><br>Correcta<br>Puntúa como 1,00<br><input type="button" value="Marcar pregunta"/>    | <p>La ventaja de la estrategia <i>ramificación y poda</i> frente a <i>vuelta atrás</i> es que la primera genera las soluciones posibles al problema mediante ...</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input checked="" type="radio"/> a. Las otras dos opciones son verdaderas. ✓</li> <li><input type="radio"/> b. ... 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.</li> <li><input type="radio"/> c. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Pregunta 5</b><br>Correcta<br>Puntúa como 1,00<br><input type="button" value="Marcar pregunta"/>    | <p>La complejidad en el peor de los casos de un algoritmo de <i>vuelta atrás</i> ...</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input checked="" type="radio"/> a. ... es exponencial con el número de decisiones a tomar. ✓</li> <li><input type="radio"/> b. ... puede ser exponencial con el número de alternativas por cada decisión.</li> <li><input type="radio"/> c. ... puede ser polinómica con el número de decisiones a tomar.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Pregunta 6</b><br>Correcta<br>Puntúa como 1,00<br><input type="button" value="Marcar pregunta"/>    | <p>Al resolver el problema del viajante de comercio mediante <i>vuelta atrás</i> y asumiendo un grafo de <math>n</math> vértices totalmente conexo, ¿cuál de estas es una buena cota pesimista al iniciar la búsqueda?</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. Se multiplica <math>n</math> por la distancia de la arista más corta que nos queda por considerar.</li> <li><input type="radio"/> b. Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las <math>n</math> aristas más cortas.</li> <li><input checked="" type="radio"/> c. Se resuelve el problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido. ✓</li> </ul>                                                                                                                                                                                                                                                                                                                                   |
| <b>Pregunta 7</b><br>Correcta<br>Puntúa como 1,00<br><input type="button" value="Marcar pregunta"/>    | <p>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 para como cota para limitar la búsqueda en un algoritmo de <i>vuelta atrás</i>.</p> <p>¿Qué tipo de cota sería?</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. No se trataría de ninguna poda puesto que es posible que esa heurística no encuentre una solución factible.</li> <li><input type="radio"/> b. Una cota pesimista.</li> <li><input checked="" type="radio"/> c. Una cota optimista. ✓</li> </ul>                                                                                                                                                        |
| <b>Pregunta 8</b><br>Correcta<br>Puntúa como 1,00<br><input type="button" value="Marcar pregunta"/>    | <p>En los algoritmos de <i>ramificación y poda</i> ...</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida.</li> <li><input checked="" type="radio"/> b. Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima. ✓</li> <li><input type="radio"/> c. Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Pregunta 9</b><br>Incorrecta<br>Puntúa como 1,00<br><input type="button" value="Marcar pregunta"/>  | <p>En los algoritmos de <i>ramificación y poda</i>, ¿el valor de una cota pesimista es menor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. En general sí, si se trata de un problema de maximización, aunque en ocasiones ambos valores pueden coincidir.</li> <li><input type="radio"/> b. Sí, siempre es así.</li> <li><input checked="" type="radio"/> c. En general sí, si se trata de un problema de minimización, aunque en ocasiones ambos valores pueden coincidir. X</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Pregunta 10</b><br>Incorrecta<br>Puntúa como 1,00<br><input type="button" value="Marcar pregunta"/> | <p>Di cuál de estas tres soluciones a problemas de optimización no comporta, en el peor caso, tener que considerar <math>O(n!)</math> posibilidades.</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. La solución de <i>ramificación y poda</i> al problema de la asignación de <math>n</math> tareas a <math>n</math> 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.</li> <li><input type="radio"/> b. La solución al problema de buscar un árbol que cubre todos los vértices de un grafo de <math>n</math> vértices de forma que el coste es mínimo (<i>minimum spanning tree</i>).</li> <li><input checked="" type="radio"/> c. La solución de <i>vuelta atrás</i> al problema del viajante de comercio (<i>travelling salesman problem</i>), o sea, el de encontrar un ciclo hamiltoniano de coste mínimo en un grafo conexo de <math>n</math> vértices donde cada arista tiene un coste asignado. X</li> </ul> |
| <b>Pregunta 11</b><br>Incorrecta<br>Puntúa como 1,00<br><input type="button" value="Marcar pregunta"/> | <p>En la estrategia de <i>ramificación y poda</i> ...</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input checked="" type="radio"/> a. ... cada nodo tiene su propia cota pesimista, la cota optimista sin embargo, es común para todos los nodos. X</li> <li><input type="radio"/> b. ... cada nodo tiene su propia cota pesimista y también su propia cota optimista.</li> <li><input type="radio"/> c. ... cada nodo tiene su propia cota optimista, la cota pesimista sin embargo, es común para todos los nodos.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Pregunta 12</b><br>Correcta<br>Puntúa como 1,00<br><input type="button" value="Marcar pregunta"/>   | <p>Cuando se resuelve usando un algoritmo de <i>ramificación y poda</i> un problema de <math>n</math> decisiones, en el que siempre hay como mínimo dos opciones para cada decisión, ¿cuál de las siguientes complejidades en el caso peor es la mejor que nos podemos encontrar?</p> <p>Seleccione una:</p> <ul style="list-style-type: none"> <li><input type="radio"/> a. <math>O(n^2)</math></li> <li><input checked="" type="radio"/> b. <math>O(2^n)</math> ✓</li> <li><input type="radio"/> c. <math>O(n!)</math></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

Pregunta 1

Correcta

Puntuación 1.00

 Marcar pregunta

En los algoritmos de *ramificación y poda*, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

Seleccione una:

- 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, nunca es así.

**Pregunta 2**

Sin contestar

Puntuación 1,00

 Marcar pregunta

**En los algoritmos de ramificación y poda ...**

Seleccione una:

- a. Una cota pesimista es necesariamente un valor inalcanzable, de no ser así no está garantizado que no se eliminen todos los factibles.
- b. Una cota optimista es necesariamente un valor insuperable, de no ser así se podría pasar el nodo que conduce a la solución óptima.
- c. Una cota pesimista es el beneficio esperado de cualquier nodo factible.

Pregunta 3

Correcto

Puntuación 1,00

 Marcar pregunta

En el esquema de vuelta atrás, los mecanismos de poda basados en la mejor solución hasta el momento ...

Seleccione una:

- a. Las dos anteriores son verdaderas.
- b. .... garantizan que no se va a explorar nunca todo el espacio de soluciones posibles.
- c. ... pueden eliminar soluciones parciales que son factibles. ✓

**Pregunta 4**

Incorrecto

Puntuación 0,00

 Marcar

pregunta

La estrategia de ramificación y poda necesita cotas pesimistas ...

Seleccione una:

- a. ... para decidir el orden de visita de los nodos del árbol de soluciones.
- b. ... para determinar si una solución es factible. 
- c. ... sólo si se usa para resolver problemas de optimización.

Pregunta 5

Incorrecta

Puntuación 1,00

Puntaje

Preguntas

La complejidad en el mejor de los casos de un algoritmo de ramificación y poda...

Seleccione una:

- a. ... suele ser polinómica con el número de alternativas por cada decisión X
- b. ... es siempre exponencial con el número de decisiones a tomar.
- c. ... puede ser polinómica con el número de decisiones a tomar.

**Pregunta 6**

Incorrecta

Puntuación 0,00

 Marcar pregunta

Si para resolver un mismo problema usamos un algoritmo de ramificación y poda y lo modificamos mínimamente para convertirlo en un algoritmo de vuelta atrás, ¿qué cambiamos realmente?

Seleccione una:

- a. Sería necesario comprobar si las soluciones son factibles o no puesto que ramificación y poda sólo genera nodos factibles. 
- b. Provocamos que las cotas optimistas pierdan eficacia.
- c. Cambiamos la función que damos a la cota pesimista.

Pregunta 7

Incorrecto

Puntaje como 1,00



Preguntas

¿Para qué sirven las cotas pesimistas en *ramificación y poda*?

Seleccione una:

- a. Para tener la certeza de que la cota optimista está bien calculada.
- b. Para descartar nodos basándose en la preferencia por algún otro nodo ya completado. X
- c. Para descartar nodos basándose en el beneficio esperado.

Pregunta 1

Correctas

Puntaje como 1,00

 Marcar preguntas

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 en el caso peor es la mejor que nos podemos encontrar?

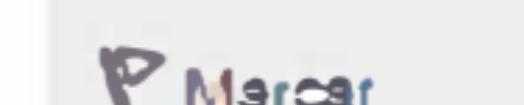
Seleccione una:

- a.  $O(2^n)$  ✓
- b.  $O(n^2)$
- c.  $O(n!)$

Pregunta 2

Correctas

Puntaje como 1,00



Pregunta

Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

Seleccione una:

- 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 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. ✓
- c. Se resuelve el resto del problema usando un algoritmo voraz que añade cada vez al camino el vértice más cercano al último añadido.

Pregunta 3

Correcto

Puntúa como 1,00

 Marcar

Preguntas

La estrategia de *ramificación y poda* genera las soluciones posibles al problema mediante ...

Seleccione una:

- a. ... un recorrido en profundidad del árbol que representa el 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 anchura del árbol que representa el espacio de soluciones.

Pregunta 4

Correctas

Puntúa como 1,00

 Marcar

Preguntas

Se desea obtener todas las permutaciones de una lista compuesta por  $n$  elementos. ¿Qué esquema es el más adecuado?

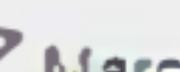
Seleccione una:

- a. Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.
- b. Vuelta atrás, para este problema no hay un esquema más eficiente. ✓
- c. Ramificación y poda, puesto que con buenas funciones de cota es más eficiente para este problema que vuelta atrás.

Pregunta 5

Incorrecto

Puntaje como 1,00

 Marcar pregunta

En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

Seleccione una:

- a. es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione. 
- b. podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.
- c. una estrategia voraz puede ser la única alternativa.

**Pregunta 6**  
Correcta

Puntúa como 1.00

 **Marcar**  
pregunta

En los **algoritmos de ramificación y poda**, ¿el valor de una cota pesimista es mayor que el valor de una cota optimista? (entendiendo que ambas cotas se aplican sobre el mismo nodo)

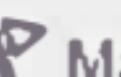
Seleccione una:

- a. En general sí, si se trata de un problema de **minimización**, aunque en ocasiones ambos valores pueden coincidir. ✓
- b. En general sí, si se trata de un problema de **maximización**, aunque en ocasiones ambos valores pueden coincidir.
- c. No, nunca es así.

**Pregunta 9**

Sin contestar

Puntaje como 1.00

 Marcar pregunta

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?

Seleccione una:

- a. Aprovechamos mejor las cotas optimistas.
- b. Cambiamos la función que damos a la cota pesimista.
- c. La comprobación de las soluciones factibles: en *ramificación y poda* no es necesario puesto que sólo genera nodos factibles.

Pregunta 10

Correcta

Puntaje como 1.00

 Marcar pregunta

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 geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino según su distancia geográfica. Este algoritmo voraz, ¿serviría como cota pesimista?

Seleccione una:

- a. No, ya que no asegura que se encuentre una solución factible. ✓
- b. Sí, puesto que la distancia geográfica asegura que otra solución mejor no es posible.
- c. No, ya que en algunos casos puede dar distancias menores que la óptima.

Pregunta 11

Correcto

Puntúa como 1,00

 Marcar

Pregunta

En los algoritmos de ramificación y poda ...

Seleccione una:

- a. El uso de cotas pesimistas sólo resulta eficaz cuando se dispone de una posible solución de partida.
- b. Una cota optimista es necesariamente un valor alcanzable, de no ser así no está garantizado que se encuentre la solución óptima.
- c. Una cota optimista es necesariamente un valor insuperable, de no ser así se podría podar el nodo que conduce a la solución óptima. ✓

Pregunta 12

Incorrecto

Puntaje como 1,00

Marcas

preguntas

Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila:

Seleccione una:

- a. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos. X
- b. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- c. El valor óptimo de la mochila continua correspondiente.

Pregunta 7

La estrategia de ramificación y poda genera las soluciones posibles al problema mediante ...

Correcto

Puntuación 1,00

 Marcar

Preguntas

Selección una:

- a. ... un recorrido en anchura del árbol que representa el espacio de soluciones.
- b. ... un recorrido en profundidad del árbol que representa el espacio de soluciones.
- c. ... un recorrido guiado por estimaciones de las mejores ramas del árbol que representa el espacio de soluciones. ✓

**Pregunta 8**  
Incorrecta

Puntuación 1,00

 Marcar pregunta

Decid cuál de estas tres es la cota optimista que poda más eficientemente cuando se usa la estrategia de vuelta atrás para resolver el problema de la mochila:

Seleccione una:

- a. El valor óptimo de la mochila continua correspondiente.
- b. El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos. 
- c. El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.

Pregunta 9

Correcta

Puntuación 1,00

▼ Marcar pregunta

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 geográfica (en línea recta) entre cada par de ciudades. Para limitar la búsqueda en un algoritmo de vuelta atrás, se utiliza la solución de un algoritmo voraz basado en moverse en cada paso a la ciudad, de entre las posibles según el mapa de carreteras, que esté más cercana al destino en línea recta.

¿Qué tipo de cota sería?

Seleccione una:

- a. Sería una cota pesimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible. ✓
- b. Sería una cota optimista siempre que se tenga la certeza de que esa aproximación encuentra una solución factible.
- c. Ninguna de las otras dos opciones.

**Pregunta 10**

Correcta

Puntuación 1,00

Marcar pregunta

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 en el caso peor es la mejor que nos podemos encontrar?

Selecciona una:

- a.  $O(2^n)$
- b.  $O(n!)$
- c.  $O(n^2)$

Pregunta 11

Correcta

Puntuación 1.00

Marcar pregunta

Al resolver el problema del viajante de comercio mediante vuelta atrás, ¿cuál de estas cotas optimistas se espera que padezca mejor el árbol de búsqueda?

Seleccione una:

- a. 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.
- 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 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.

Pregunta 12

Incorrecta

Puntuación 1,00

Marcar pregunta

El problema de cortar un tubo de longitud  $n$  en segmentos de longitud entera, de manera que el precio total de sus partes sea máximo de acuerdo con una lista de precios por longitudes ...

Seleccione una:

- a. ... se debe resolver mediante un algoritmo de vuelta atrás, dado que otros algoritmos no consideran todas las posibles maneras de cortar el tubo.
- b. ... se puede resolver mediante un algoritmo de vuelta atrás pero existe una solución

---

asintóticamente mucho más eficiente.

- c. ... no se puede resolver usando un algoritmo de vuelta atrás.

Pregunta 8

Correcto

Puntúa como 1,00

Marcar

preguntas

La complejidad en el mejor de los casos de un algoritmo de vuelta atrás ...

Seleccione una:

- a. ... puede ser polinómica con el número de decisiones a tomar. ✓
- b. ... suele ser polinómica con el número de alternativas por cada decisión
- c. ... es siempre exponencial con el número de decisiones a tomar.

VERSIÓN 1

COMPSCI 220

# LA UNIVERSIDAD DE AUCKLAND

**PRIMER SEMESTRE 2013****Campus: Ciudad**

## CIENCIAS DE LA COMPUTACIÓN

### Algoritmos y estructuras de datos

**(Tiempo permitido: 40 minutos)****NOTA:**

- Ingrese su nombre e identificación de estudiante en la hoja de Teleform PRIMERO.
- ENTONCES: ¡Intente *todas las* preguntas!
- Todas las preguntas tienen UNA respuesta correcta.
- NO marque dos respuestas como correctas para la misma pregunta.
- Si cree que hay un error en una pregunta (varias respuestas correctas o ninguna respuesta correcta), seleccione la respuesta que crea que fue la correcta y comuníquese con el supervisor de la sala de examen después del examen.
- Consérve su libro de preguntas. No se marcará la escritura en el libro de preguntas.
- NO se permite el uso de calculadoras.
- ¡Buena suerte!

CONTINUADO

---

**Página 2****VERSIÓN 1**

2

COMPSCI 220

1. ¿Cuál de las siguientes funciones **no** es  $O(n^2)$ ?

- A.  $f(\text{norte}) = (\text{norte}^2 + 2)(2n + 7) - \text{norte}^3$
- B.  $f(\text{norte}) = \text{mi}^3 \text{norte}^2$
- C.  $f(n) = n(2n + 7) - 1$
- D.  $f(\text{norte}) = \text{norte}^2(\text{norte} + 7) - \text{norte}^3$
- E.  $f(n) = e^{-n}$

2. ¿Cuál de las siguientes funciones es  $\Theta(n^2)$ ?

- A.  $f(n) = n^2 \sin(15n)$
- B.  $f(n) = n(2n + 7) - 1$
- C.  $f(n) = n^3 + n$
- D.  $f(n) = n(n+1)^{2n}$
- E.  $f(n) = n^3 + n^2$

3. Si  $g(n)$  es  $O(n)$  y  $h(n)$  es  $\Theta(n^2)$ , ¿cuál de las siguientes afirmaciones sobre la función  $f(n) = g(n) + h(n)$  es cierto?

- A.  $f(n)$  es  $\Theta(n^2)$
- B.  $f(n)$  es  $\Omega(n^3)$
- C.  $f(n)$  es  $O(n)$
- D.  $f(n)$  es  $\Theta(n^3)$
- E.  $f(n)$  es  $\Theta(n)$

4. ¿Cuántas comparaciones hace mergesort al fusionar dos listas de longitud  $n$  cada una?

- A. Como máximo  $2n - 1$
- B. Como máximo  $n - 1$
- C.  $n \log n$
- D. Como máximo  $n$
- E. Exactamente  $2n - 1$

5. Un cierto algoritmo de clasificación ordena su lista de la siguiente manera: seleccione un elemento en la lista existente, luego mueva

todos los elementos con una clave más alta a la derecha del elemento seleccionado, y todos los elementos con una clave más baja a la izquierda del elemento seleccionado. Luego ordene las listas a la izquierda / derecha del elemento seleccionado, cada una por aplicando de nuevo el algoritmo de forma recursiva. Este algoritmo de clasificación se conoce como:

- Un rápido
- B. montón
- C. selección
- D. fusionar
- E. inserción

CONTINUADO

## Página 3

### VERSIÓN 1

3

COMPSCI 220

6. ¿Cuál de las siguientes afirmaciones sobre el ordenamiento por selección **no** es cierta?

- A. La clasificación por selección tiene una complejidad de tiempo promedio de  $\Omega(n^2)$
- B. La ordenación por selección tiene una complejidad de tiempo en el mejor de los casos que es mejor que la cúbica
- C. La ordenación por selección siempre toma el mismo número de comparaciones para un tamaño de entrada dado
- D. La ordenación por selección ordena listas pre ordenadas de la misma longitud en el mismo tiempo independientemente de dirección de clasificación
- E. El ordenamiento por selección tiene una complejidad de tiempo en el peor de los casos de  $\Omega(n^2 \log n)$

7. Un algoritmo tarda aproximadamente 1 segundo para una entrada de tamaño  $n = 10$ , aproximadamente 200 segundos para una entrada de tamaño  $n = 100$ , y aproximadamente 20000 segundos para  $n = 1000$ . Se observa el algoritmo la complejidad del tiempo es:

- A. numérico
- B. lineal
- C. cúbico
- D. exponencial
- E. cuadrático

8. Antes de Facebook, una clase COMPSCI220 tuvo una idea. Querían producir un libro como recuerdo, con una página para cada alumno de la clase. El libro se imprimirá y se entregará una copia a cada uno. estudiante. Suponga que el costo de impresión es proporcional a la cantidad de papel utilizado. Si es así, el costo de imprimir el libro para una clase de  $n$  estudiantes habría sido:

- A.  $O(n \log n)$
- B.  $\Theta(n^2)$
- C.  $\Theta(n^3)$
- D.  $O(n)$
- E.  $\Omega(n^3)$

9. El tipo de datos abstractos de tabla (ADT) se puede implementar como una tabla hash o como un árbol equilibrado. Para ¿Cuál de las siguientes operaciones de tabla es una tabla hash más lenta que un árbol equilibrado?

- A. Enumerar

- B. Eliminar
- C. Insertar
- D. Actualización
- E. Búsqueda

10. ¿Qué propiedad de un montón máximo permite una implementación no recursiva de heapsort?

- A. Un montón se puede almacenar simplemente en una matriz de nodos
- B. Insertar o eliminar nodos de montón es una operación  $\Theta(\log n)$
- C. La clave de valor más alto siempre está en la raíz de un montón máximo
- D. Cada nodo de montón máximo tiene un valor clave  $\geq$  los valores clave de sus nodos secundarios
- E. Un montón es un árbol binario completo

CONTINUADO

## Página 4

### VERSIÓN 1

4

COMPSCI 220

11. ¿Qué relación de recurrencia tiene la solución exacta  $T(n) = 2^n + 1$  para  $T(0) = 2$ ?

- A.  $T(n) = 2(T(n - 1) + 1)$
- B.  $T(n) = 2T(n - 1) - 1$
- C.  $T(n) = T(n - 1) + 1$
- D.  $T(n) = T(n - 1) + 2$
- E.  $T(n) = 2T(n - 1) + 1$

12. ¿Cuál de las siguientes afirmaciones sobre árboles de búsqueda m –aria **no** es verdadera?

- A. No se pueden usar cuando todos sus nodos están almacenados en la memoria.
- B. Sus nodos deben ser más grandes para que puedan contener más claves
- C. Reducen la cantidad de E / S de disco cuando se trabaja con conjuntos de datos muy grandes.
- D. Es posible que las búsquedas deban realizar comparaciones de  $m - 1$  en cada nodo
- E. Su altura es menor que la de un árbol binario con el mismo número de claves

13. Prueba un algoritmo que se sabe que exhibe complejidad de tiempo cúbico. Con un tamaño de entrada de  $n = 100$ , el algoritmo se ejecuta durante un día. A medida que duplica el tamaño de la entrada ( $n = 200$ ), el algoritmo tarda 8 días en completo. ¿Cuánto tiempo puede tardar el tamaño de entrada 1000?

- A. dos semanas
- B. poco más de un año
- C. casi tres años
- D. nueve meses
- E. tres meses

14. ¿Cuál de las siguientes no es una operación elemental?

- A. Iterando cinco veces a través de un bucle for que contiene una sola operación elemental
- B. Comprobación de si una cadena de entrada contiene la letra 'x'
- C. Incrementar un número entero en Java
- D. Comparación de dos variables booleanas para determinar la igualdad
- E. Imprimiendo la cadena 'HOLA MUNDO'

15. Para una tabla hash que usa una matriz con n elementos, ¿por qué usamos el resto  $h(k) = k \% n$  como valor hash?

- A. Esta función hash será más rápida que tomar partes de la clave k y agregarlas
- B. La división es el método básico más común utilizado en funciones hash.
- C. Es posible que queramos usar  $(k / n) \bmod n$  como nuestro decremento de doble hash
- D. Hacerlo garantiza que  $h(k)$  se encuentra dentro de la matriz hash
- E. Calcular un resto es más rápido que calcular un cociente

CONTINUADO

## Página 5

### VERSIÓN 1

5

COMPSCI 220

16. ¿Cuál de los siguientes algoritmos de clasificación es *estable* y tiene la misma complejidad para el peor, el promedio y mejores casos de sus datos de entrada?

- A. mergesort
- B. shellsort
- C. ordenación por inserción
- D. clasificación rápida
- E. heapsort

17. ¿Cuál de los siguientes pares de complejidades temporales es más difícil de diferenciar mediante la observación?

- A.  $\Theta(n)$  y  $\Theta(n \log n)$
- B.  $\Theta(2^n)$  y  $\Theta(n^2)$
- C.  $\Theta(\log n)$  y  $\Theta(n \log n)$
- D.  $\Theta(n^2)$  y  $\Theta(n \log n)$
- E.  $\Theta(n)$  y  $\Theta(n^2)$

18. Una *inversión* en una matriz de números enteros es un par de elementos de matriz a [i] y a [j] para los cuales  $i < j$  y  $a[i] > a[j]$ . ¿Cuál de las siguientes afirmaciones **no** es cierta?

- A. Una matriz ordenada no tiene inversiones
- B. Se puede eliminar exactamente una inversión si intercambiamos a [i] y a [i + 1]
- C. Intercambiar un [i] y un [i + espacio] puede eliminar más de una inversión
- D. Shellsort puede eliminar solo inversiones de espacio para cada tamaño de espacio que utiliza
- E. El desempeño de Shellsort depende de su secuencia de tamaños de espacios

CONTINUADO

---

**Página 6**

VERSIÓN 1

6

COMPSCI 220

**Página de trabajo 1**

CONTINUADO

---

**Página 7**

VERSIÓN 1

7

COMPSCI 220

**Página de trabajo 2**

VERSIÓN 1

COMPSCI 220

# LA UNIVERSIDAD DE AUCKLAND

SEGUNDO SEMESTRE, 2011

Campus: Ciudad

CIENCIAS DE LA COMPUTACIÓN

Algoritmos y estructuras de datos

(Tiempo permitido: UNA hora)

NOTA:

¡Intente todas las preguntas!  
NO se permite el uso de calculadoras.

CONTINUADO

---

**Página 2**

VERSIÓN 1

2

COMPSCI 220

1. ¿Qué algoritmos de clasificación cubiertos en este curso son comparativos, estables, implementados y tienen  
¿Tiempo de funcionamiento asintóticamente óptimo?

- A. clasificación rápida, clasificación en pilas
- B. heapsort, mergesort
- C. mergesort, quicksort
- D. ninguno
- E. ordenación por inserción, ordenación en pila

2. ¿Cuál de estos NO está en  $\Omega(n \log n)$ ?

- A.  $0.001n^2 / \lg n$
- B.  $17n \ln n$
- C.  $n^{3/4} (\ln n)^5$
- D.  $\sqrt{n}^3 \log_{10} n$
- E.  $n^2 (2 - \cos(n))$

3. Un cierto algoritmo de tiempo cuadrático utiliza 300 operaciones elementales para procesar una entrada de tamaño 10.  
¿Cuál es el número más probable de operaciones elementales que utilizará si se le da una entrada de tamaño 1000?

- A. 3000
- B. 30 000
- C. 3 000 000
- D. 300 000
- E. 300 000 000

4. Elija el enunciado verdadero más fuerte: el ordenamiento por selección hace al menos tantas comparaciones como la inserción  
clasificar

- A. para entrada con menos inversiones que el promedio
- B. en cada entrada
- C. en el peor de los casos
- D. en entrada ordenada inversa
- E. en promedio para una entrada uniformemente aleatoria

5. Un algoritmo cuyo tiempo de ejecución para el tamaño de entrada  $n$  satisface la relación de recurrencia (para  $n \geq 1$ )

$$T(n) = \frac{1}{n} (T(0) + T(1) + \dots + T(n-1)) + 5n$$

tiene tiempo de ejecución en

- A.  $\Theta(\log n)$
- B.  $\Theta(n \log n)$
- C.  $\Theta(n^{\text{norte}})$
- D.  $\Theta(n^2)$
- E.  $\Theta(n)$

CONTINUADO

---

**Página 3**

VERSIÓN 1

3

COMPSCI 220

6. Está intentando romper la implementación de clasificación de alguien, pero no puede ver su código fuente. Observas que el algoritmo se ejecuta en un tiempo similar para conjuntos de datos ordenados y desordenados, y el rendimiento es bueno en los grandes. Los artículos con la misma clave a veces cambiarán su orden relativo. ¿Cuál es más probable que utilicen el algoritmo de clasificación?

- A. clasificación rápida
- B. ordenación por selección
- C. heapsort
- D. ordenación por inserción
- E. mergesort

7. El tiempo de ejecución del siguiente fragmento de código es  $\Theta(f(n))$ . ¿Qué es  $f(n)$ ?

```
para (int i = 1; i <n; i = i * 2) {
 para (int j = 0; j <n; j++) {
 si (i == j) {
 para (int k = 0; k <n; k++) {
 // Haz algo elemental
 }
 }
 demás{
 // Haz otra cosa elemental
 }
 }
}
```

- A.  $n^2 \log n$
- B.  $n \log n$
- C.  $n$
- D.  $n (\log n)^2$
- E.  $n^2$

8. El número mínimo posible de comparaciones realizadas en el peor de los casos mediante una clasificación basada en comparaciones. algoritmo para ordenar una lista de tamaño 4 es

- A. 5
- B. 4
- C. 6
- D. 8

CONTINUADO

**Página 4**

VERSIÓN 1

4

COMPSCI 220

9. Sabes que el algoritmo A se ejecuta en un tiempo exponencial  $\Theta(2^n)$ . Si su computadora puede procesar la entrada de tamaño 1000 en un año usando una implementación de este algoritmo, aproximadamente qué tamaño de entrada Espera poder resolver en un año con una computadora 1000 veces más rápido?

- A. 32 000
- B. 1 010
- C. 1 500
- D. 1 002
- E. 1 000 000

10. La recurrencia  $T(n) = T(n/2) + n^2$ ,  $T(1) = 0$  tiene una solución que es de orden exacto

- A.  $Un$
- B.  $n^2$
- C.  $2^n$
- D.  $n \log n$
- E.  $\log n$

11. Después de insertar los elementos 4, 6, 1, 3, 5, 2 en ese orden en un max-heap binario inicialmente vacío, ¿qué es el hijo correcto de la raíz?

- A. 1
- B. 4
- C. 5
- D. 2
- E. 3

12. ¿Cuál de las siguientes entradas hará la implementación estándar de quicksort (siempre elija el elemento más a la izquierda como pivote) ¿hace el menor número de comparaciones?

- A. 4,3,1,2,6,5,7
- B. 7,6,5,4,3,2,1
- C. 1,2,3,4,5,6,7
- D. 1,3,5,7,6,4,2
- E. 2,4,6,7,5,3,1

13. Si el algoritmo A tiene complejidad de tiempo cuadrático y se ejecuta durante 1 segundo en un problema de tamaño 100, aproximadamente ¿Cuánto tiempo esperaría que se tardara en resolver un problema del tamaño de un millón?

- A. 3 años
- B. 300 años
- C. 12 días
- D. 4 meses
- E. 3 horas

CONTINUADO

**Página 5**

VERSIÓN 1

5

COMPSCI 220

14. Elija la declaración FALSE:  $7 \lg n + 5n + n \lg \lg n + 3n \ln n$  es

- A.  $\Omega(n)$
- B.  $\Theta(n \log n)$
- C.  $\Omega(n^2)$
- D.  $O(n^2)$
- E.  $O(n \log_2 n)$

15. Para un árbol, \*\*\* se define como el máximo de todas las profundidades de los nodos. Que es \*\*\*?

- A. profundidad total
- B. ancho
- C. orden
- D. longitud del camino interno
- E. altura

16. Un cierto algoritmo de tiempo cúbico usa 30 operaciones elementales para procesar una entrada de tamaño 10. ¿Qué es  
¿Cuál es el número más probable de operaciones elementales que utilizará si se le da una entrada de tamaño 1000?

- A. 3000
- B. 30 000
- C. 300 000
- D. 3 000 000
- E. 30 000 000

17. La recurrencia  $T(n) = T(n/2) + n^2$ ,  $T(1) = 0$  tiene una solución exacta (para una potencia de 2):

- A.  $2n - 2$
- B.  $(4n^2 - 4)/3$
- C.  $\lg n$
- D.  $(4^n - 4)/3$
- E.  $2^n - 2$

18. Considere las siguientes declaraciones. ¿Cuáles son las verdaderas?

- (i) Si  $f(n) = n^2$ ,  $g(n) = (1 + (-1)^n)n$ , entonces  $g(n)$  es  $O(f(n))$
- (ii) Si  $f(n) = n$ ,  $g(n) = (1 + (-1)^n)n^2$ , entonces  $g(n)$  es  $\Omega(f(n))$
- (iii) Si  $f(n) = n^2 \log_4(n)$ ,  $g(n) = (5n^2 + 1)(\lg n + \lg \lg n)$ , entonces  $f(n)$  es  $\Theta(g(n))$

- A. yo, ii
- B. ii, iii
- C. ninguno
- D. todos
- E. i, iii

CONTINUADO

**Página 6**

VERSIÓN 1

6

COMPSCI 220

19. Se le presenta un algoritmo de clasificación desconocido \*\*\* - sort. El algoritmo parece ejecutarse en una velocidad constante en conjuntos de datos aleatorios y ordenados, y no se ralentiza notablemente a medida que aumentar en tamaño. Dos elementos cualesquiera que tengan la misma clave de búsqueda no cambiarán su orden relativo. ¿Cuál de las siguientes opciones es más probable que sea \*\*\*?

- A. Una selección
- B. inserción
- C. rápido
- D. fusionar
- E. montón

20. ¿Cuántos de los 5 árboles binarios que se muestran están completos?



- A. 1
- B. 5
- C. 2
- D. 3
- E. 0

CONTINUADO

---

**Página 7**

VERSIÓN 1

7

COMPSCI 220

Página de trabajo 1

CONTINUADO

---

**Página 8**

VERSIÓN 1

8

COMPSCI 220

Página de trabajo 2

VERSIÓN 1

COMPSCI 220

# LA UNIVERSIDAD DE AUCKLAND

SEGUNDO SEMESTRE 2012  
Campus: Ciudad

## CIENCIAS DE LA COMPUTACIÓN

Algoritmos y estructuras de datos

(Tiempo permitido: 50 minutos)

### NOTA:

- Ingrese su nombre e identificación de estudiante en la hoja de Teleform PRIMERO.
- ENTONCES: ¡Intente las 20 preguntas!
- Todas las preguntas tienen UNA respuesta correcta.
- NO marque dos respuestas como correctas para la misma pregunta.
- Si cree que hay un error en una pregunta (varias respuestas correctas o ninguna respuesta correcta), seleccione la respuesta que crea que fue la correcta y contáctenos después de la prueba.
- Conserve su libro de preguntas. No se marcará la escritura en el libro de preguntas.
- NO se permite el uso de calculadoras.
- ¡Buena suerte!

---

Página 2

VERSIÓN 1

2

COMPSCI 220

1. Sabes que el algoritmo A se ejecuta en un tiempo exponencial  $\Theta(2^n)$ . Si su computadora puede procesar la entrada de tamaño 100 en un año usando una implementación de este algoritmo, aproximadamente qué tamaño de entrada Espera poder resolver en un año con una computadora 1000 veces más rápido?

- A. 110
- B. 1 00 000
- C. 1024
- D. 10000
- E. 1500

2. Suponga que la función  $f(n)$  es  $\Theta(n)$  y que la función  $g(n)$  es  $O(n \log n)$ . Entonces el producto función  $h(n) = f(n) \cdot g(n)$  es ...

- A.  $\Omega(n^2 \log n)$
- B.  $O(n^2 \log n)$
- C.  $\Theta(n^2 \log n)$
- D.  $\Theta(n^2)$
- E.  $O(n^2)$

3. ¿Cuál de las siguientes entradas hace que el ordenamiento por selección realice el mayor número de comparaciones?

- A. 2, 3, 5, 4, 1
- B. 3, 5, 4, 1, 2
- C. 1, 2, 3, 4, 5
- D. todas las demás opciones son correctas
- E. 5, 4, 3, 2, 1

4. Se le presenta un algoritmo de clasificación desconocido \*\*\* - sort. El algoritmo parece ejecutarse en una velocidad constante en conjuntos de datos aleatorios y ordenados, y no se ralentiza notablemente a medida que aumentar en tamaño. Dos elementos cualesquiera que tengan la misma clave de búsqueda no cambiarán su orden relativo. ¿Cuál de las siguientes opciones es más probable que sea \*\*\*?

- A. Una selección
- B. montón
- C. fusionar
- D. inserción
- E. rápido

CONTINUADO

5. El estúpido algoritmo de ordenación funciona de la siguiente manera para ordenar una lista en su lugar: Comenzando desde el principio de la lista, escanee hasta que se encuentren dos elementos sucesivos que están en el orden incorrecto. Cambie esos elementos y vuelva a el principio. El algoritmo finaliza cuando se llega al final de la lista.

El tiempo de ejecución en el peor de los casos para una lista de tamaño  $n$  es de orden:

- A.  $\Omega(n)$
- B.  $n^2$
- C.  $n \log n$
- D.  $n^3$
- E.  $n^n$

6. ¿Cuál de las siguientes cadenas de bits hace que un analizador Lempel-Ziv encuentre un patrón conocido que se superpone? con el inicio de la repetición del patrón desde la posición de análisis actual?

- A. 0100110 ...
- B. 1011010 ...
- C. 0110100 ...
- D. 0101011 ...
- E. 0011000 ...

7. Escribe un programa en el que el usuario ingresa los nombres de uno en uno, con un nombre vacío que indica el fin de entrada. ¿Cuál de las siguientes estructuras de datos es la más adecuada para almacenar dicha lista de nombres y salida en reversa?

- A. árbol binario
- B. pila
- C. cola de prioridad
- D. montón
- E. cola

8. Elija la declaración FALSE:  $7 \lg n + 5n + n \lg \lg n + 3n \ln n$  es

- A.  $\Omega(n^2)$
- B.  $O(n \log_2 n)$
- C.  $\Theta(n \log n)$
- D.  $O(n^2)$
- E.  $\Omega(n)$

9. Si el algoritmo A tiene una complejidad de tiempo cuadrático y se ejecuta durante 1 segundo en un problema de tamaño 200, aproximadamente ¿Cuánto tiempo esperaría que se tardara en resolver un problema del tamaño de dos millones?

- A. 3 horas
- B. 300 años
- C. 12 días
- D. 3 años
- E. 4 meses

CONTINUADO

10. La matriz [9,5,6,3,2,15] representa un árbol binario completo de la forma habitual. Hazlo en un montón filtrando hacia arriba o hacia abajo un solo elemento. ¿Cuál es la matriz resultante?

- A. [3,9,6,5,2,15]
- B. [15,5,6,3,2,9]
- C. [6,5,9,3,2,15]
- D. [9,5,15,3,2,6]
- E. [15,5,9,3,2,6]

11. Dada una matriz ordenada de 11 enteros [5 9 13 17 22 29 35 41 47 51 55], se le pedirá que utilice la búsqueda binaria para buscar el valor 41. Los punteros bajo y alto de la búsqueda se establecen inicialmente en 0 y 10. ¿Qué son sus valores después de dos ciclos (es decir, reducciones a la mitad del tamaño de la búsqueda)?

- A. bajo = 5, alto = 7
- B. bajo = 6, alto = 7
- C. bajo = 6, alto = 8
- D. bajo = 7, alto = 7
- E. bajo = 5, alto = 8

12. Está intentando romper la implementación de clasificación de alguien, pero no puede ver su código fuente. Observas que el algoritmo se ejecuta en un tiempo similar para conjuntos de datos ordenados y desordenados, y el rendimiento es bueno en los grandes. Los artículos con la misma clave a veces cambiarán su orden relativo. ¿Cuál es más probable que utilicen el algoritmo de clasificación?

- A. clasificación rápida
- B. ordenación por selección
- C. ordenación por inserción
- D. mergesort
- E. heapsort

13. Considere un analizador sintáctico LZ77 con un tamaño de ventana (tanto para la ventana de búsqueda hacia atrás como para la ventana de anticipación) de 4 bits que operan en una cadena binaria. Suponiendo que el analizador finaliza la búsqueda en un paso tan pronto como se encuentra la coincidencia de longitud máxima, ¿cuál es el mayor número de comparaciones de bits realizadas en un paso?

- A. 16
- B.  $\Theta(n)$
- C. 10
- D. 24
- E. 13

CONTINUADO

14. Después de insertar los elementos 8, 12, 2, 6, 10, 3 en ese orden en un max-heap binario inicialmente vacío, ¿Cuál es el hijo correcto de la raíz?

- A. 2
- B. 8
- C. 6
- D. 10
- E. 3

15. ¿Cuál de las siguientes entradas hace que mergesort realice el mayor número de comparaciones?

- A. 2, 1, 4, 3
- B. 1, 2, 3, 4
- C. 4, 3, 2, 1
- D. 1, 3, 2, 4
- E. todas las demás opciones son correctas

16. Considere las siguientes declaraciones. ¿Cuáles son las verdaderas?

- (i) Si  $f(n) = n^2$ ,  $g(n) = (1 + (-1)^n)n$ , entonces  $g(n)$  es  $O(f(n))$
- (ii) Si  $f(n) = n$ ,  $g(n) = (1 + (-1)^n)n^2$ , entonces  $g(n)$  es  $\Omega(f(n))$
- (iii) Si  $f(n) = n^2 \log 4(n)$ ,  $g(n) = (5n^2 + 1)(\lg n + \lg \lg n)$ , entonces  $f(n)$  es  $\Theta(g(n))$

- A. ninguno
- B. ii, iii
- C. i, iii
- D. yo, ii
- E. todos

17. Un cierto algoritmo de tiempo cúbico usa 30 operaciones elementales para procesar una entrada de tamaño 10. ¿Qué es  
¿Cuál es el número más probable de operaciones elementales que utilizará si se le da una entrada de tamaño 1000?

- A. 3 000 000
- B. 30 000 000
- C. 30 000
- D. 300 000
- E. 3000

18. Considere todos los posibles algoritmos de ordenación basados en comparaciones aplicados para ordenar una lista de tamaño 4. Para cada algoritmo a, sea  $W(a)$  el número de comparaciones en el peor de los casos para ordenar dicha lista. El mínimo posible valor de  $W(a)$  es:

- A. 4
- B. 6
- C. 5
- D. 8
- E. 3

CONTINUADO

19. Sea la función  $f(n) \Theta(n^2)$ . Entonces la función  $f(n)$  es simultáneamente ...

- A.  $O(n)$  y  $\Omega(n^3)$
- B.  $O(n^3)$  y  $\Omega(n^3)$
- C.  $O(n^3)$  y  $\Omega(n^2)$
- D.  $O(n^2)$  y  $\Omega(n^3)$
- E.  $O(n)$  y  $\Omega(n^2)$

20. El tiempo de ejecución del siguiente fragmento de código es  $\Theta(f(n))$ . ¿Qué es  $f(n)$ ?

```
para (int i = 1; i <n; i = i * 2) {
 para (int j = 0; j <n; j++) {
 si (i == j) {
 para (int k = 0; k <n; k++) {
 // Haz algo elemental
 }
 }
 demás{
 // Haz otra cosa elemental
 }
 }
}
```

- A.  $n^2$
- B.  $n(\log n)^2$
- C.  $n^2 \log n$
- D.  $n$
- E.  $n \log n$

CONTINUADO

---

## Página 7

VERSIÓN 1

7

COMPSCI 220

Página de trabajo 1

CONTINUADO

---

**Página 8**

VERSIÓN 1

8

COMPSCI 220

Página de trabajo 2

Preguntas:

1. Decid cuál de estas tres estrategias proveerá la cota pesimista más ajustada al valor óptimo de la mochila discreta:

- (a) Completar las decisiones restantes basándose en la mejor solución voraz que pueda encontrarse para los restantes objetos y espacio disponible de la mochila.
- (b) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.
- (c) Asumir que ya no se van a coger más objetos.



2. ¿Cuál de estos problemas tiene una solución eficiente utilizando *programación dinámica*?

- (a) El problema del cambio.
- (b) La mochila discreta con pesos y valores reales positivos.
- (c) El problema de la asignación de tareas.



$O(1^n)$

3. 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?

$$f(n) = 1 + f(n-2)$$

$$f(n) = 1 + f(n-2) = 2 + f(n-4) = 3 + f(n-6)$$

$$f(n) = i + f(n-2i)$$

$$n-2i = 0 \Rightarrow n = 2i$$

$$i = \frac{n}{2}$$

$$f(n) = i + f(n-2i)$$

4. 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) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
- (c) 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.

5. ¿En un esquema de RyP, ¿podrían coincidir los valores obtenidos por las cotas pesimista y optimista de un nodo cualquiera?

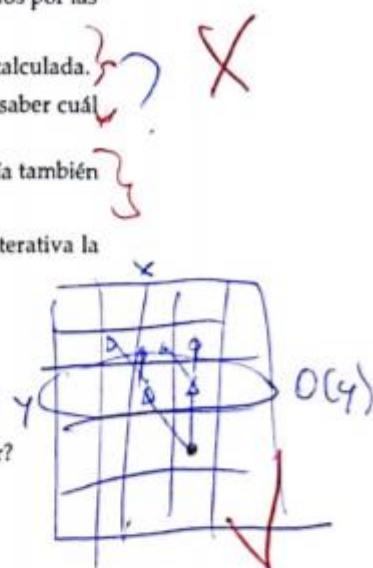
- (a) No, en tal caso una de las cotas (o ambas) estaría mal calculada.
- (b) Si, pero habría que seguir completando el nodo para saber cuál es la mejor solución que puede obtenerse de él.
- (c) Sí, en tal caso el valor obtenido por las cotas coincidiría también con el mejor valor que puede obtenerse de ese nodo.

6. Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int y, int x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
```

¿Cuál es la mejor complejidad espacial que se puede conseguir?

- (a)  $O(x \cdot y)$
- (b)  $O(y)$
- (c)  $O(x)$



7. ¿Cuál de las siguientes estrategias de búsqueda es más apropiada en un esquema de vuelta atrás?

- (a) Explorar primero los nodos con mejor valor hasta el momento en la función que se pretende optimizar.
- (b) Ninguna de las otras dos estrategias es compatible con el esquema de vuelta atrás.
- (c) Explorar primero los nodos con mejor cota optimista.

8. 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)  $\Omega(n^2) \subset \Omega(n^3)$  F
- (b)  $\Theta(n^2) \subset \Theta(n^3)$  F
- (c)  $O(n^2) \subset O(n^3)$  V



9. De las siguientes afirmaciones marca la que es verdadera.

- (a) En un esquema de vuelta atrás, las cotas pesimistas no tienen sentido si lo que se pretende es obtener todas las soluciones factibles.
- (b) El esquema de vuelta atrás no es compatible con el uso conjunto de cotas pesimistas y optimistas.
- (c) Las cotas pesimistas no son compatibles con un esquema de vuelta atrás.



10. El algoritmo de ordenación *Quicksort* divide el problema en dos subproblemas. ¿Cuál es la complejidad temporal asintótica de realizar esa división?

- (a)  $\Theta(\log n)$
- (b)  $\Theta(n \log n)$
- (c) Ninguna de las otras dos opciones es correcta.

$\Theta(n)$  ✓

11. Se desea ordenar una lista enlazada de  $n$  elementos adaptando el 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)  $\Theta(n^2)$
- (b) Ninguna de las otras dos opciones es cierta.
- (c)  $\Theta(n \log n)$

✓

12. El esquema de vuelta atrás ...

- (a) Garantiza que encuentra la solución óptima a cualquier problema de selección discreta.
- (b) Se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado. *No, solo con finito*
- (c) Las otras dos opciones son ambas verdaderas.

✓ principio de optimidad

13. Queremos resolver mediante vuelta atrás el problema de las 8 reinas (colocar 8 reinas en un tablero de ajedrez de manera que no se maten mutuamente). Una buena cota optimista permitiría:

- X (a) Muy probablemente, explorar menos nodos.
- (b) Muy probablemente, resolver el problema de forma más rápida.
- (c) No es aplicable este tipo de podas a este problema.

✓

14. Se desea resolver el problema de la potencia enésima ( $x^n$ ), asumiendo que  $n$  es par y que se utilizará la siguiente recurrencia:  $\text{pot}(x, n) = \text{pot}(x, n/2) * \text{pot}(x, n/2)$ ; ¿Qué estrategia resulta ser más eficiente en cuanto al coste temporal?

- (a) En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.
- (b) Un algoritmo recursivo con memoización.
- (c) Divide y vencerás.

✓

pero resuelto  
no de los lados  
ya no tienes que  
resolver el otro

15. Dada la siguiente función (donde  $\max(a, b) \in \Theta(1)$ ):

```
float exa(vector<float>&v, vector<int>&p, int P, int i)
{
 float a, b;
 if (i >= 0) {
 if (p[i] <= P)
 a = v[i] + exa(v, p, P - p[i], i - 1);
 else a = 0;
 b = exa(v, p, P, i - 1);
 }
 return max(a, b);
}
return 0;
```



Marcad la opción correcta.

- (a) La complejidad temporal en el mejor de los casos es  $\Omega(n^2)$
- (b) La complejidad temporal en el peor de los casos es  $O(n^2)$
- (c) La complejidad temporal en el peor de los casos es  $O(2^n)$

→ se expone como  
en arbol  
?

16. Si  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$  entonces ...

+ orden u  
de arriba  
ej:  $g(n) = n$ ,  $f(n) = n^2$

- (a) ...  $f(n) \in \Theta(g(n))$
- (b) ...  $f(n) \in \Omega(g(n))$
- (c) ...  $f(n) \in O(g(n))$



17. ¿Qué nos proporciona la media aritmética entre el coste temporal asintótico (o complejidad temporal) en el peor caso y el coste temporal asintótico en el mejor caso?

- (a) En general, nada de interés.
- (b) El coste temporal asintótico en el caso medio. NO
- (c)  $T(n) \in \Theta(n)$  y  $T(n) \in \Omega(n^2)$



18. 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.

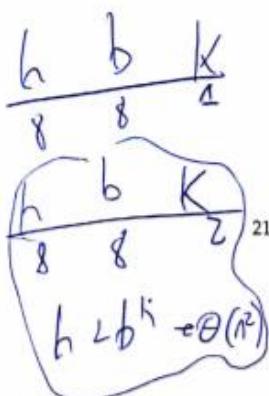
- (a)  $O(n^2) \subset O(2^{\log_2 n})$  F
- (b)  $\Omega(n^2) \subset \Omega(n)$  V
- (c)  ~~$n \cdot n \log_2 n \in \Omega(n + n \log_2 n)$~~  N ?

19. 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) Ramificación y poda, puesto que con buenas funciones de cota es más eficiente que vuelta atrás.
- (c) Divide y vencerás, puesto que la división en sublistas se podría hacer en tiempo constante.



h b K  
8 8 3



20. Sea la siguiente relación de recurrencia

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 8T\left(\frac{n}{8}\right) + g(n) & \text{en otro caso} \end{cases}$$

Si  $T(n) \in \Theta(n^2)$ , ¿en cuál de estos tres casos nos podemos encontrar?

- (a)  $g(n) = n^3$
- (b)  $g(n) = n$
- (c)  $g(n) = n^2$



21. En un problema de optimización, si el dominio de las decisiones es un conjunto infinito,

- (a) podremos aplicar el esquema vuelta atrás siempre que se trate de un conjunto infinito numerable.
- (b) una estrategia voraz puede ser la única alternativa.
- (c) es probable que a través de programación dinámica se obtenga un algoritmo eficaz que lo solucione.



???

22. En un algoritmo de optimización resuelto mediante ramificación y poda ¿Podría encontrarse la solución óptima sin haber alcanzado nunca un nodo hoja?

- (a) Si, pero esto solo podría ocurrir si se hace uso de cotas pesimistas.
- (b) Si, esto puede ocurrir incluso si no se hace uso de cotas pesimistas.
- (c) No, los nodos hojas son los nodos completados y por lo tanto hay que visitar al menos uno de ellos para almacenarlo como la mejor solución hasta el momento.

✓?

23. Si  $f \notin O(g_1)$  y  $f \in O(g_2)$  entonces siempre se cumplirá:

- (a)  $f \notin O(\max(g_1, g_2))$  **No**
- (b)  $f \in \Omega(g_1 + g_2)$
- (c)  $f \in \Omega(\min(g_1, g_2))$  \*



24. 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. *o lo contrario*
- (c) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.



25. ¿Qué complejidad se obtiene a partir de la relación de recurrencia  $T(n) = 9T(n/3) + n^3$  con  $T(1) = O(1)$ ?

- (a)  $O(n \log n)$
- (b)  $O(n^3 \log n)$
- (c)  $O(n^3)$

h b K  
9 3 3  
 $h < b^K = \Theta(n^3)$

26. Se pretende resolver el problema del viajante de comercio (*travelling salesman problem*) mediante el esquema de vuelta atrás, ¿cuál de los siguientes valores se espera que se comporte mejor para decidir si un nodo es prometedor?

- (a) El valor que se obtiene de multiplicar  $k$  por el peso de la arista más corta de entre las restantes, donde  $k$  es el número de ciudades que quedan por visitar.
- (b) La suma de los pesos de las aristas que completan la solución paso a paso visitando el vértice más cercano al último visitado.
- (c) La suma de los pesos de las  $k$  aristas restantes más cortas, donde  $k$  es el número de ciudades que quedan por visitar. *optimista → la mejor parte*

Problema de  
minimización

???)

27. Se pretende aplicar la técnica memoización a la siguiente función recursiva:

```
int f(int m, int n) {
 if(m <= n) return 1;
 return m * f(m-1,n) + n;
}
```

$\rightarrow m = 1$  tanto temporal como espacial

¿Qué complejidades temporal y espacial cabe esperar de la función resultante?

- (a) Ninguna de las otras dos opciones es correcta.
  - (b)  $O(n - m)$ , tanto espacial como temporal.
  - (c)  $O(m - n)$ , tanto espacial como temporal.
28. Dado un problema de minimización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente pequeño?
- (a) Que se podría explorar más nodos de los necesarios.
  - (b) Que se podría explorar menos nodos de los necesarios.
  - (c) Que se podría podar el nodo que conduce a la solución óptima.

M2  
hay espacial?

29. ¿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)  $\Theta(n \cdot m)$
- (b)  $\Theta(n)$
- (c)  $\Theta(n^2)$

30. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...

- (a) ... no se puede usar para resolver problemas de optimización.
- (b) ... debe recorrer siempre todo el árbol.
- (c) ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.

31. Si el coste temporal de un algoritmo es  $T(n)$ , ¿cuál de las siguientes situaciones es imposible?

- (a)  $T(n) \in O(n)$  y  $T(n) \in \Theta(n)$  ✓
- (b)  $T(n) \in \Omega(n)$  y  $T(n) \in \Theta(n^2)$  ✓
- (c)  $T(n) \in \Theta(n)$  y  $T(n) \in \Omega(n^2)$  F

32. 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) Las otras dos opciones son ambas falsas.
- (c) Siempre es mayor o igual que la mejor solución posible alcanzada.

33. 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 en el mejor de los casos es  $\Omega(n)$  → este algoritmo ordena
- (b) La complejidad temporal exacta es  $\Theta(n^2)$
- (c) La complejidad temporal en el mejor de los casos es  $\Omega(1)$

34. Con respecto a la complejidad espacial de los algoritmos de ordenación Quicksort, Heapsort y Mergesort ...

- (a) Mergesort y Heapsort tienen complejidad espacial lineal con el tamaño del vector a ordenar, la de Quicksort es constante.
- (b) Mergesort tiene complejidad espacial lineal con el tamaño del vector a ordenar, la de los otros dos es constante.
- (c) Las complejidades espaciales de todos ellos son lineales con el tamaño del vector a ordenar.

???

35. ¿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) Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.
- (c) Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.

???

36. Un algoritmo recursivo basado en el esquema divide y vencerás...

- (a) ... alcanza su máxima eficiencia cuando el problema de tamaño  $n$  se divide en  $a$  problemas de tamaño  $n/a$ . Falso
- (b) Las otras dos opciones son ambas verdaderas. Falso
- (c) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.

✓

37. ¿Qué tienen en común los algoritmos de ordenación Quicksort y Mergesort.

- (a) El número de llamadas recursivas que hacen en el mejor de los casos.
- (b) La complejidad temporal de la combinación de las soluciones parciales.
- (c) La complejidad temporal de la división en subproblemas.

X

38. Dado el problema de las torres de Hanoi resuelto mediante divide y vencerás, ¿cuál de las siguientes relaciones de recurrencia expresa mejor su complejidad temporal para el caso general, siendo  $n$  el número de discos?

- (a)  $T(n) = T(n-1) + n$
- (b)  $T(n) = 2T(n-1) + 1$
- (c)  $T(n) = 2T(n-1) + n$

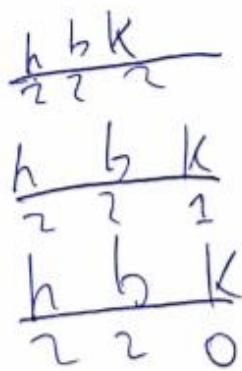
???

$$a) f(n) = \underset{i=2}{\dots} + f(n-1) = n + \underset{i=2}{\dots} + f(n-2) = n + \underset{i=2}{\dots} + (n-1) + f(n-3)$$

$$f(n) = \underset{i=2}{\dots} + \sum_{i=2}^n$$

$$c) f(n) = n + 2f(n-2) = n + 2n - 2 + 4f_{\underset{2}{\dots}}^{n-2}$$

i=3



39. La siguiente relación de recurrencia expresa la complejidad de un algoritmo recursivo, donde  $g(n)$  es una función polinómica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + g(n) & \text{en otro caso} \end{cases}$$

Dí cuál de las siguientes afirmaciones es cierta:

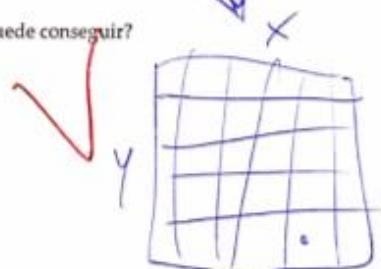
- (a) Si  $g(n) \in \Theta(n^2)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación mediante inserción binaria.
- (b) Si  $g(n) \in \Theta(n)$  la relación de recurrencia representa la complejidad temporal del algoritmo de ordenación *Mergesort*.
- (c) Si  $g(n) \in \Theta(1)$  la relación de recurrencia representa la complejidad temporal del algoritmo de búsqueda dicotómica.

40. Se pretende implementar mediante programación dinámica iterativa la función recursiva:

```
int f(int y, int x){ // suponemos y >= x
 if (x==0 || y==x) return 1;
 return f(y-1, x-1) + f(y-1, x);
}
```

¿Cuál es la mejor complejidad temporal que se puede conseguir?

- (a)  $O(x \cdot y)$
- (b)  $O(y)$
- (c)  $O(x)$



30. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...

- (a) ... no se puede usar para resolver problemas de optimización.
- (b) ... debe recorrer siempre todo el árbol.
- (c) ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.

31. Si el coste temporal de un algoritmo es  $T(n)$ , ¿cuál de las siguientes situaciones es imposible?

- (a)  $T(n) \in O(n)$  y  $T(n) \in \Theta(n)$  ✓
- (b)  $T(n) \in \Omega(n)$  y  $T(n) \in \Theta(n^2)$  ✓
- (c)  $T(n) \in \Theta(n)$  y  $T(n) \in \Omega(n^2)$  F

32. 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. No
- (b) Las otras dos opciones son ambas falsas.
- (c) Siempre es mayor o igual que la mejor solución posible alcanzada. No

33. 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 en el mejor de los casos es  $\Omega(n)$  → el algoritmo ordena
- (b) La complejidad temporal exacta es  $\Theta(n^2)$
- (c) La complejidad temporal en el mejor de los casos es  $\Omega(1)$