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 soluci on 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(n2)
= 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 pode 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 reconsidere la decision ya tomada anteriormente respecto a la selecci on de un elemento a la vista de la decisi on 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) esm as 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 funci on polin omica: T(n) =1 si n <= 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<=1///2T(n/2) + g(n) en otro caso Si T(n) pertenece O(n), ¿en cusal de estos tres casos nos podemos encontrar?

= g(n) = 1
~ g(n) = n2
~ 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<=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(n2)

¿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 n2 pertenece a O(n^3)

Sea g(n) = (sumatorio [desde i=0] hasta K) 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 × 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 venceras

¿Cual es la diferencia principal entre una solucion de vuelta atras y una solucion de ramificaion 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 0(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 venceras ...

= ...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 necesarya 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; for(i = 0; i <n ; i++) 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_i 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 , ,,,, , n + sumatorio de t(j) n > 1, 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 cojunto de funciones dadas , al secunecia 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 + nlogn pertenece omega(n)
~ n + nlogn pertenece promedio(n)

si el coste temporal de un algoritmo es T8n) , ¿cual 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(logn)
~ 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
¿cual 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

cuando la 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 venceras
= programacion dinamica
~ ramificacion y poda

un problema de tamaño n puede transformarse en tiempo O(n^2) en nueve de tamaño n/3 por otro lado la solucion al problema cuando la talla es 1 requiere un tiempo constante, ¿cual de estas clases de coste temporal asintotico es las mas ajustada?

~ O(nlogn)
= O(n^2 logn)
~ O(n^2)

indica cual de las siguientes expresiones es falsa

~ promedio(n/2) = promedio(n)
~ promedio(n) C O(n)
= promedio(n) C 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 logn)

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 demas opciones son falsas
~ siempre coincidira con la complejidad temporal de las instancias que estan en el caso base del algoritmo recursiva

cuando se resuelve usando backtracking un problema de n decisiones en el que siempre hay como minimo 2 opciones para cada decision, ¿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 venceras

~ Quicksort
~ Mergesort
= el algoritmo de Prim

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

~ la complejidad temporal de la mejor solucion posible al problema es 0(n!)
~ 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 Omega(n^2)

Sea la siguiente relacion de recurrencia T(n) = 1 si n<=1 //////[2T (n/2) + g(n) en otro caso Si T(n) pertenece 0(n cuadrado), ¿en cual de estos tres casos nos podemos encontrar?

~ g(n) = n
= g(n) = n cuadrado
~ g(n) = 1

¿Cual de estos tres problemas de optimizacion no tiene, o no se le conoce, una solucion voraz (greedy) que es optima?

= el problema de la mochila discreta
~ el problema de la mochila continua o con fraccionamiento
~ el arbol de cobertura de coste minimo de un grafo conexo

Un algoritmo recursivo basado en el esquema divide y venceras…

~ … nunca tendra una complejidad exponencial
= … sera mas eficiente cuanto mas equitativa sea la division en subproblemas
~ las dos anteriores son ciertas

Un problema de tamaño n puede transformarse en tiempo 0(n cuadrado) en otro de tamaño n – 1. Por otro lado, la solucion al problema cuando la talla es 1 requiere un tiempo constante, ¿cual de estas clases de coste temporal asintotico es la mas ajustada?

~ 0(2^n)
= 0(n^3)
~ 0(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 mas pequeña que se le puede presentar
~ las otras dos opciones son ciertas
= … es una funcion 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 atras, ¿cual de estas cotas optimistas se espera que pode 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

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

La version de Quicksort que utiliza como pivote el elemento del vector que ocupa la primera posicion…

~ … 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

Si f(n) pertenece 0(n^3), ¿puede pasar que f(n) pertenece 0(n^2)?

~ no, porque n al cubo "no incrementa" 0(n al cuadrado)
~ solo para valores bajos de n
= es perfectamente posible, ya que 0(n cuadrado) C 0 (n al cubo)

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 se 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 enel 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 0(n) y f(n) pertenece omega(1)
= f(n) pertenece mejor caso(n cuadrado) y f(n) pertenece 0(n)
~ f(n) pertenece 0(n) y f(n) pertenece 0(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 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

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 0(n log n) y este es 0(n)
= solo si d log d > k n (donde k es una constante que depende de la implementacion)
~ 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 atras y lo modificamos minimamente para convertirlo en un algoritmo de ramificacion y poda, ¿que cambiamos realmente?

~ cambiamos la funcion que damos a la cota pesimista
= el algoritmo puede aprovechar mejor las cotas optimistas
~ la comprobacion de las soluciones factibles: en ramificacion 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, ¿cual sera la complejidad temporal del mejor algoritmo que pueda existir?

~ 0 (n cuadrado)
= 0 (n)
~ 0 (raiz cuadrada n)

Cuando la descomposicion recursiva de un problema da lugar a subproblemas de tamaño similar, ¿que esquema promete ser mas apropiado?

= programacion dinamica
~ divide y venceras, siempre que se garantice que los subproblemas no son del mismo tamaño
~ el metodo 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 meidante programacion dinamica
= ... en la solucion ingenua se resuelve muchas veces un numero relativamente pequeño 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, cij 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 subconjuto del conjuto de elementos que optimiza una determinada funcion), ¿cual de estas tres cosas es imposible que ocurra?

~ que el algoritmo no encuentre ninguna solucion
= que se reconsidere 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 venceras" 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 venceras
~ 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 pode 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 cij 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 entres 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 geograficasde 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 uy 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 funcion 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 polinomica con el numero de decisiones a tomar
~ suele ser polinomica con el numero de alternativas por cada decisión

un algoritmo recursivo basado en divide y venceras

~ nunca tendra una complejidad exponencial
= sera mas eficiente cuanto mas equitativa sea la division en subproblemas
~ las dos anteriores son correctas

cuando se usa un algoritmo voraz para abordar la resolucion de un problema de optimizacion por seleccion directa(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 se reconsidere 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

cuando la descomposicion recursiva de un problema da lugar a subproblemas de tamaño similar, que esquema promete ser mas apropiado?

= programacion dinamica
~ divide y venceras , siempre que se garantice que los subproblemas no son del mismo tamaño
~ voraz

sea la siguiente relacion de recurrencia
t(n) + 1 si n<= 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 solucion 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 solucion eficiente que siga el esquema de programación 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

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 teste
~ una cota superior para el valor optimo
= una cota inferior para el valor optimo que a veces puede ser igual a este

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 algorimo voraz basado en el valor especifico 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 solucion al problema cuando la talla es 1 requiere un tiempo constante, ¿cual de estas clases de coste temporal asintotico es la mas ajustada?

~ O(2^n)
= O(n^3)
~ O(n^2)

Sea la siguiente relacion de recurrencia :::: T(n)= 1 si n<=1 ////// [2T (n/2) + g(n) en otro caso. Si T(n) e O(n), ¿en cual de estos tres casos nos podemos encontrar?

~ g(n)=n^2
= las otras dos opciones son ambas ciertas
~ g(n) = log n

¿Que se deduce de f(n) y g(n) se se cumple lim n... infinito [f(n)/g(n)] = K, con K distinto 0?

~ g(n) pertenece 0[0(f(n)] pero f(n) "no incremento" 0[g(n)]
= f(n) pertenece 0[g(n)] y g(n) "incremento 0`f(n)]
~ f(n) pertenece 0[g(n)] pero g(n) "no incremento" 0[f(n)]

¿Cual seria la complejidad temporal de la siguiente funcion tras aplicar programacion dinamica?

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

~ promedio(n x 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: pot (x,n) = pot (x,n/2) * 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 ¿cual 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...

~ ... 0(log n)
~ ... mejor(n^2)
= ... 0(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 ¿cual 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 mejor(g1) y f pertenece mejor(g2) entonces

~ f pertenece mejor(g1 * g2)
= f pertenece mejor(g1 + g2)
~ f no pertenece mejor[min (g1, g2)]

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) = 0(1) ?

~ 0(n log n)
= 0(n^3 log n)
~ 0(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;
                }
        }
```

¿cual es su complejidad temporal asintotica?

~ 0(n log n)
~ 0(n "al cuadrado")
= 0(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), ¿cual 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 0(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 garantia de que sea una buena estrategia de busqueda
~ si, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse segun 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 venceras...

~ ... nunca tendra un coste temporal asintotico (o complejidad temporal) exponencial
~ las otras dos opciones son ambas verdaderas
= ... alcanza su maxima 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 numero de caminos distintos desde la casilla incial (1,1) hasta la casilla (n,m) y para ello se aplica el esquema programacion dinamica para obtener un algoritmo lo mas eficiente posible en cuanto a complejidad temporal y espacial, ¿cuales serian ambas complejidades?

~ temporal promedio[max(n,m)] y espacial promedio[max (n,m)]
~ temporal promedio(n x m) y espacial promedio(n x m)
= temporal promedio(n x m) y espacial promedio[min (n,m)]

Dado el problema del laberinto con tres movimientos, se pretende conocer la longitud del camino de salida mas corto. Para ello se aplica el esquema voraz con un criterio de seleccion 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 ultimo, si este tampoco es posible, se escoge el movimiento Sur. ¿Que se puede decir del algoritmo obtenido?

~ que en realidad no es un algoritmo voraz pues las decisiones que se toman no deberian reconsiderarse
~ que es un algoritmo voraz pero sin garantia de solucionar el problema
= que en realidad no es un algoritmo voraz pues el criterio de seleccion no lo e

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

= ... no recorre todo el arbol si hay manera de descartar subarboles que representan conjuntos de soluciones no factibles
~ ...debe recorrer siempre todo el arbol
~ ...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 conjuto de cotas pesimistas y optimistas
~ las cotas pesimistas no son compatibles con un esquem ade vuelta atras

El esquema voraz...

~ puede que no encuentre una solucion pero si lo hace se garantiza que es optima
= las otras dos opciones son ambas falsas
~ garantiza encontrar una solucion a cualquier problema, aunque puede que no sea optima

Cuando la 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?

= programacion dinamica
~ ramifiacion y poda
~ divide y venceras

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 programacion dinamica. En cuanto a la complejidad temporal, ¿cual es la mejora de la version recursiva con memoizacion frente a la recursiva ingenua que se obtiene a partir del esquema divide y venceras?

~ de una complejidad cuadratica que se obtendria con la ingenua se reduciria a lineal con la de memoizacion
~ la mejora no esta garantizada puesto que la version recursiva con memoizacion podria ser peor que la obtenida a partir del esquema divide y venceras
= de una complejidad exponencial que se obtendria con la ingenua se reduciria a polinomica con la de memoizacion

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 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

~ 0(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, ¿cual 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 venceras, 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 mejorarlo transformandolo en un algoritmo de programacion dinamica iterativa, pero se le añade memoizacion. ¿podria ser que el algoritmo iterativo p evalue 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(g1) y f pertenece a O(g2) entonces NO siempre se cumplira

~ f pertenece omega(min(g1,g2))
~ f pertenece O(max(g1,g2))
= f pertenece omega(g1+g2)

Que tiene que valer k en la relacion de recurrencia T(n)= 1 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 n<1 /// 1+bT(n-1) 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-->infinito (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 x n d + n 2/d)
~ O(n d x 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 solución 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 ,¿cual es la complejidad temporal asintotica en funcion 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 deberia 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 maximizacion utilizando ramificacion y poda, y para ello se dispone de 3 cuotas optimistas ¿Cual deberiamos utilizar para reducir la cantidad de nodos a explorar?

~ La que obtenga valores mas elevados. PUEDE SER ESTA
= La que obtenga valores mas pequeños. YO PONDRIA ESTA
~ La que se acerque mas a una heuristica voraz que obtenga una solucion aproximada.

Tenemos un vector ordenado y queremos comprobar si contiene un elemento dado ¿ Cual sera la complejidad temporal mas 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<= i < v.size(); i < j <= v.size()
Se quiere reducir la complejidad temproal usano programacion dinamica iterativa, cual seria la complejidad espacial

~ cubica
= cuadratica
~ exponencial

Tratandose de un esquema general para resolver problemas de minimizacion ¿ que falta en el hueco?

Solution BB (Problem p)
if(????????????)

= n.optimistic_b() <= pb
~ n.optimistic_b() >= pb
~ n.pesimistic_b() <= pb

Tratandose de un esquema general para resolver problemas de maximizacion ¿ que falta en el hueco?

Solution BB (Problem p)
if(????????????)

~ n.optimistic_b() <= pb
= n.optimistic_b() >= pb
~ n.pesimistic_b() <= pb

La programacion dinamica...

= Las otras dos opciones son ciertas. Correcta
~ ... normalmente se usa para resolver problemas de optimizacion 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 optimizacion con dominios continuos pero probablemente pierda su eficacia ya que puede disminuir drasticamente el numero de subproblemas repetidos.

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

= O(y)
~ O(y2)
~ 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(n2) a O(nlogn), 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 Θ(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 Θ(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(y2)
= 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(n2) a O(nlogn), 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.