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

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

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

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

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 pode mejor el árbol de búsqueda?

   (a) Se multiplica k por la distancia de la arista más corta que nos queda por considerar, donde k es el número de saltos que nos quedan por dar.

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

   [(c)] Se ordenan las aristas restantes de menor a mayor distancia y se calcula la suma de las k aristas más cortas, donde k es el número de saltos que nos quedan por dar.

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 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.

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(XXXXXXX));
}
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. ¿cual 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 mas corto entre dos ciudades. Para ello se dispone de una tabla con la distancia entre los pares de ciudades en los que hay carreteras o un valor centinela (por ejemplo, -1) si no hay, por lo que para ir de la ciudad inicial a la final es posible que haya que pasar por varias ciudades. También se conocen las coordenadas geográficas de cada ciudad y por tanto la distancia geométrica (en línea recta) entre cada par de ciudades. Se pretende acelerar la búsqueda de un algoritmo de *ramificación y poda* priorizando los nodos vivos (ciudades) que estén a menor distancia geográfica de la ciudad objetivo.

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

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 reconsidere la decisión ya tomada anteriormente respecto a la selección de un elemento a la vista de la decisión que se debe tomar en el instante actual.

    (c) Que la solución no sea la óptima.

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.