Respuestas para la modalidad **2**

1. En ausencia de cotas optimistas y pesimistas, la estrategia de vuelta atrás...
   - (a) ... no se puede usar para resolver problemas de optimización.
   - [(b)] ... no recorre todo el árbol si hay manera de descartar subárboles que representan conjuntos de soluciones no factibles.
   - (c) ... debe recorrer siempre todo el árbol.
2. El esquema voraz ...
   - (a) Garantiza encontrar una solución a cualquier problema, aunque puede que no sea óptima.
   - (b) Puede que no encuentre una solución pero si lo hace se garantiza que es óptima.
   - [(c)] Las otras dos opciones son ambas falsas.
3. Dado un problema de maximización resuelto mediante un esquema de ramificación y poda, ¿qué ocurre si la cota optimista resulta ser un valor excesivamente elevado?
   - (a) Que se podría podar el nodo que conduce a la solución óptima.
   - [(b)] Que se podría explorar más nodos de los necesarios.
   - (c) Que se podría explorar menos nodos de los necesarios.
4. ¿Qué se deduce de $f(n)$ y $g(n)$ si se cumple $\lim_{n\to\infty}(f(n)/g(n)) = k$, con $k \neq 0$?
   - (a) $f(n) \in O(g(n))$ pero $g(n) \notin O(f(n))$
   - (b) $g(n) \in O(f(n))$ pero $f(n) \notin O(g(n))$
   - [(c)] $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
5. ¿Qué ocurre si la cota pesimista de un nodo se corresponde con una solución que no es factible?
   - (a) Nada especial, las cotas pesimistas no tienen por qué corresponderse con soluciones factibles.
   - [(b)] Que el algoritmo sería incorrecto pues podría descartarse un nodo que conduce a la solución óptima.
   - (c) Que el algoritmo sería más lento pues se explorarían más nodos de los necesarios.
6. Un algoritmo recursivo basado en el esquema *divide y vencerás*...
   - [(a)] ... alcanza su máxima eficiencia cuando el problema de tamaño n se divide en a problemas de tamaño n/a.
   - (b) ... nunca tendrá un coste temporal asintótico (o complejidad temporal) exponencial.
   - (c) Las otras dos opciones son ambas verdaderas.

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

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

   Marcad la opción correcta.

   [(a)] La complejidad temporal en el peor de los casos es $O(2^n)$

   (b) La complejidad temporal en el mejor de los casos es $\Omega(n^2)$

   (c) La complejidad temporal en el peor de los casos es $O(n^2)$

8. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1, 1)$ hasta la casilla (n, m) y para ello se aplica un esquema de divide y vencerás.¿Cuál sería la recurrencia apropiada para el caso general?

   (a) $nc(n, m) = nc(n - 1, m) * nc(n, m - 1) * nc(n - 1, m - 1)$

   (b) Ninguna de las otras dos recurrencias se corresponde con un esquema de divide y vencerás.

   [(c)] $nc(n, m) = nc(n - 1, m) + nc(n, m - 1) + nc(n - 1, m - 1)$

9. Se desea resolver el problema de la potencia enésima (x^n), asumiendo que n es par y que se utilizará la siguiente recurrencia: `pot(x,n) = pot(x,n/2) * pot(x,n/2);` ¿Qué esquema resulta ser más eficiente en cuanto al coste temporal?

   (a) Divide y vencerás.

   (b) En este caso tanto programación dinámica como divide y vencerás resultan ser equivalentes en cuanto a la complejidad temporal.

   [(c)] Programación dinámica.

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

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

    (b) Ninguna de las otras dos opciones es cierta.

    [(c)] $\Theta(n \log n)$

11. Dada la siguiente función:

```
int exa (vector <int>& v){
   int j, i=1, n=v.size();

   if (n>1) do{
      int x = v[i];
      for (j=i; j >0 and v[j-1] >x; j--)
         v[j]=v[j-1];
      v[j]=x;
      i++;
   } while (i<n);
   return 0;
}
```

Marcad la opción correcta.

[(a)] La complejidad temporal en el mejor de los casos es $\Omega(n)$

(b) La complejidad temporal en el mejor de los casos es $\Omega(1)$

(c) La complejidad temporal exacta es $\Theta(n^2)$

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

(a) Pila

[(b)] Cola

(c) Cola de prioridad

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

(a) Siempre es mayor o igual que la mejor solución posible alcanzada.

[(b)] Las otras dos opciones son ambas falsas.

(c) Asegura un ahorro en la comprobación de todas las soluciones factibles.

14. Dado el problema del laberinto con tres movimientos, se pretende conocer la longitud del camino de salida más corto. Para ello se aplica el esquema voraz con un criterio de selección que consiste en elegir primero el movimiento *Este* siempre que la casilla sea accesible. Si no lo es se descarta ese movimiento y se prueba con *Sureste* y por último, si este tampoco es posible, se escoge el movimiento *Sur*. ¿Qué se puede decir del algoritmo obtenido?

[(a)] Que en realidad no es un algoritmo voraz pues el criterio de selección no lo es.

(b) Que en realidad no es un algoritmo voraz pues las decisiones que se toman no deberían reconsiderarse.

(c) Que es un algoritmo voraz pero sin garantía de solucionar el problema.

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

    (a) $T(n) \in O(n)$ y $T(n) \in \Theta(n)$

    (b) $T(n) \in \Omega(n)$ y $T(n) \in \Theta(n^2)$

    [(c)] $T(n) \in \Theta(n)$ y $T(n) \in \Omega(n^2)$

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

    (a) $O(n^3)$

    (b) $O(n \log n)$

    [(c)] $O(n^3 \log n)$

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

    [(a)] El valor de la mochila continua correspondiente .

    (b) El valor de la mochila discreta que se obtiene usando un algoritmo voraz basado en el valor específico de los objetos.

    (c) El valor de una mochila que contiene todos los objetos aunque se pase del peso máximo permitido.

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

    [(a)] En el esquema de vuelta atrás no se pueden definir estrategias de búsqueda.

    (b) Explorar primero los nodos con mejor cota optimista.

    (c) Explorar primero los nodos que están más completados.

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

    (a) No, la cota optimista sólo se utiliza para determinar si una *n-tupla* es prometedora.

    [(b)] Sí, aunque no es una garantía de que sea una buena estrategia de búsqueda.

    (c) Sí, en el caso de que se ordene la lista de nodos vivos, siempre debe hacerse según el criterio de la cota optimista.

20. El esquema de vuelta atrás ...

    (a) Las otras dos opciones son ambas verdaderas.

    [(b)] Garantiza que encuentra la solución óptima a cualquier problema de selección discreta.

    (c) Se puede aplicar a cualquier tipo de problema aunque el coste temporal es elevado.

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

    (a) Divide y vencerás.

    [(b)] Programación dinámica.

    (c) Ramificación y poda.

22. De las siguientes afirmaciones marca la que es verdadera.

    (a) Las cotas pesimistas no son compatibles con un esquema de vuelta atrás.

    [(b)] En un esquema de vuelta atrás, las cotas pesimistas no tienen sentido si lo que se pretende es obtener todas las soluciones factibles.

    (c) El esquema de vuelta atrás no es compatible con el uso conjunto de cotas pesimistas y optimistas.

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

    (a) n^2

    [(b)] $n \log n$

    (c) $n \log^2 n$

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

    [(a)] Las otras dos estrategias son ambas válidas.

    (b) Suponer que en adelante todas las casillas del laberinto son accesibles.

    (c) Suponer que ya no se van a realizar más movimientos.

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

    [(a)] Sí, en caso de existir con este esquema siempre se puede encontrar un camino de salida

    (b) No, para garantizar que se encuentra un camino de salida hay que aplicar métodos de búsqueda exhaustiva como vuelta atrás o ramificación y poda.

    (c) No, con este esquema se puede conocer el número total de caminos distintos que conducen a la salida pero no se puede saber la composición de ninguno de ellos.

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

    (a) Lo más adecuado sería usar una técnica de ramificación y poda ya que es muy importante el orden en el que se exploran las soluciones parciales.

    [(b)] El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.

    (c) Lo más importante es conseguir una cota pesimista adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.

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

    [(a)] ... $O(n)$.

    (b) ... $O(\log n)$.

    (c) ... $\Omega(n^2)$.

28. Dada la siguiente función:

```
int exa (vector <int>& v) {
   int i,sum=0, n=v.size();

   if (n>0){
      int j=n;
      while (sum<100){
         j=j/2;
         sum=0;
         for (i=j;i<n;i++)
            sum+=v[i];
         if (j==0) sum=100;
      }
      return j;
   }
   else return -1;
}
```

    Marcad la opción correcta.

    (a) La complejidad temporal en el mejor de los casos es $\Omega(1)$

    [(b)] La complejidad temporal en el mejor de los casos es $\Omega(n)$

    (c) La complejidad temporal exacta es $\Theta(n \log n)$

29. Dada la siguiente función:

```
int exa (string & cad, int pri, int ult){

   if (pri>=ult)
         return 1;
   else
      if (cad[pri]==cad[ult])
         return exa(cad, pri+1, ult-1);
      else
         return 0;
}
```

¿Cuál es su complejidad temporal asintótica?

[(a)] $O(n)$
(b) $O(\log n)$
(c) $O(n^2)$

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

(a) $n + n \log n \in \Omega(n)$
(b) $O(2^{\log n}) \subset O(n^2)$
[(c)] $\Theta(n) \subset \Theta(n^2)$

31. ¿Cuál sería la complejidad temporal de la siguiente función tras aplicar programación dinámica?

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

(a) $\Theta(n^2)$
(b) $\Theta(nm)$
[(c)] $\Theta(n)$

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

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

33. ¿Qué nos proporciona la media entre el coste temporal asintótico (o complejidad temporal) en el peor caso y el coste temporal asintótico en el mejor caso?

    (a) El coste temporal promedio.

    (b) El coste temporal asintótico en el caso medio.

    [(c)] Nada de interés.

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

    [(a)] ... pueden eliminar vectores que representan posibles soluciones factibles.

    (b) ... garantizan que no se va a explorar todo el espacio de soluciones posibles.

    (c) Las otras dos opciones son ambas verdaderas.

35. Sea la siguiente relación de recurrencia

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

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

    (a) $g(n) = \log n$

    (b) $g(n) = \sqrt{n}$

    [(c)] Las otras dos opciones son ambas ciertas.

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

    (a) $T(n) = 2T(n-1) + n$

    (b) $T(n) = T(n-1) + n$

    [(c)] $T(n) = 2T(n-1) + 1$

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

    (a) $f \notin \Omega(\text{mín}(g_1, g_2))$

    (b) $f \in \Omega(g_1 \cdot g_2)$

    [(c)] $f \in \Omega(g_1 + g_2)$

38. Dado el problema del laberinto con tres movimientos, se desea saber el número de caminos distintos desde la casilla inicial $(1, 1)$ hasta la casilla (n, m) y para ello se aplica un esquema de programación dinámica. En cuanto a la complejidad temporal, ¿cuál es la mejora de la versión recursiva con memoización frente a la recursiva ingenua que se obtiene a partir del esquema divide y vencerás?

    [(a)] De una complejidad exponencial que se obtendría con la ingenua se reduciría a polinómica con la de memoización.

    (b) De una complejidad cuadrática que se obtendría con la ingenua se reduciría a lineal con la de memoización.

    (c) La mejora no está garantizada puesto que la versión recursiva con memoización podría ser peor que la obtenida a partir del esquema divide y vencerás.

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

    [(a)] Temporal $\Theta(n \times m)$ y espacial $\Theta(\text{mín}\{n, m\})$

    (b) Temporal $\Theta(\text{máx}\{n, m\})$ y espacial $\Theta(\text{máx}\{n, m\})$

    (c) Temporal $\Theta(n \times m)$ y espacial $\Theta(n \times m)$

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

    [(a)] un vector de booleanos.

    (b) un par de enteros que indiquen los cortes realizados y el valor acumulado.

    (c) una tabla que indique, para cada posición donde se va a cortar, cada uno de los posibles valores acumulados.