Modalidad 1

Preguntas:

1. La función γ de un número semientero positivo (un número es semientero si al restarle 0.5 es entero) se define como:

```
double gamma( double n ) { // Se asume n>=0.5 y n-0.5 entero
  if( n == 0.5 )
    return sqrt(PI);
  return n * gamma( n -1);
}
```

¿Se puede calcular usando programación dinámica iterativa?

(a) Sí.

(b) No, ya que el índice del almacén sería un número real y no entero.

(c) No, ya que no podríamos almacenar los resultados intermedios en el almacén.

2. ¿Cuál 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?

   (a) Exponencial con el número de objetos a tratar.

   (b) Polinómica con el número de objetos a tratar, siempre que se utilice programación dinámica.

   (c) Ninguna de las otras dos opciones son ciertas.

3. Un algoritmo que calcula una función recursivamente tiene coste prohibitivo y se decide mejorarlo transformándolo en un algoritmo de programación dinámica iterativa, pero antes se modifica un poco para añadirle memoización. Podría ser que el algoritmo iterativo evalúe la función más veces que el recursivo con memoización?

   (a) No, ambos evalúan la función el mismo número de veces.

   (b) No, el recursivo evalúa la función muchas más veces.

   (c) Podría ser, por ejemplo, como ocurre en el caso de la mochila discreta con pesos enteros.

4. Los algoritmos de programación dinámica hacen uso...

   (a) ...de que la solución óptima se puede construir añadiendo el componente óptimo de los restantes, uno a uno.

   (b) ...de que se puede ahorrar esfuerzo guardando los resultados de esfuerzos anteriores.

   (c) ...de una estrategia trivial consistente en examinar todas las soluciones posibles.

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

6. Asumiendo que n es par, las siguientes recurrencias matemáticas obtienen el valor de la potencia enésima (x^n):

   $\text{pot}(x,n) = \text{pot}(x,n/2) \times \text{pot}(x,n/2)$

   $\text{pot}(x,n) = n \times \text{pot}(x,n-1)$

   donde $\text{pot}(x,0) = 1$ en ambos casos ¿Cuál de las siguientes afirmaciones es cierta?

   (a) La primera recurrencia resultará ser la más eficiente siempre que se utilice la programación dinámica recursiva para su implementación.

   (b) Ambas recurrencias son equivalentes en cuanto a complejidad temporal sin importar que se haga uso de programación dinámica recursiva o de divide y vencerás.

   (c) La segunda recurrencia resulta ser la más eficiente siempre que se utilice el esquema divide y vencerás para su implementación.

7. Con respecto al tamaño del problema, ¿Cuál es el orden de complejidad temporal asintótica de la siguiente función? (asumimos que A es una matriz cuadrada)

```
void traspuesta( mat& A){
   for( int i = 1; i <A.n_rows; i++ )
      for( int j = 0; j <i ; j++ )
         swap( A(i,j), A(j,i) );
}
```

   (a) constante

   (b) lineal

   (c) cuadrático

8. Tenemos un conjunto de n enteros positivos y queremos encontrar el subconjunto de tamaño m de suma mínima.
   (a) Lo más adecuado sería usar una técnica de ramificación y poda, aunque en el peor caso el coste temporal asintótico (o complejidad temporal) sería exponencial.
   (b) Para encontrar la solución habría que probar con todas las combinaciones posibles de m enteros, con lo que la técnica de ramificación y poda no aporta nada con respecto a vuelta atrás.
   (c) Una técnica voraz daría una solución óptima.
9. Si $f \notin O(g_1)$ y $f \in O(g_2)$ entonces NO siempre se cumplirá:
   (a) $f \in \Omega(\text{mín}(g_1, g_2))$
   (b) $f \in O(\text{máx}(g_1, g_2))$
   (c) $f \in \Omega(g_1 + g_2)$
10. Qué tiene que valer k en la relación de recurrencia

$$T(n) = \begin{cases} 1 & n \leq 1 \\ n^k + 2T(\frac{n}{2}) & n > 1 \end{cases}$$

    para que $T(n) = n\log(n)$?
   (a) 0
   (b) 1
   (c) $\log(n)$
11. 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.
12. En el problema del viajante de comercio (*travelling salesman problem*) queremos listar todas las soluciones factibles.
   (a) Lo más adecuado sería usar una técnica de ramificación y poda ya que es muy importante el orden en el que se exploran las soluciones parciales.
   (b) El orden en el que se exploran las soluciones parciales no es relevante; por ello, la técnica ramificación y poda no aporta nada con respecto a vuelta atrás.
   (c) Lo más importante es conseguir una cota pesimista muy adecuada. Las diferencias entre ramificación y poda y vuelta atrás son irrelevantes en este caso.
13. ¿Cuál de las siguientes relaciones de recurrencia es la del algoritmo *mergesort*?
   (a) $T(n) = n + 2T(n/2)$ para $n > 1$ y $T(n) = 1$ para $n < 1$
   (b) $T(n) = n + T(n/2)$ para $n > 1$ y $T(n) = 1$ para $n < 1$
   (c) $T(n) = n + T(n-1)$ para $n > 1$ y $T(n) = 1$ para $n < 1$
14. Los algoritmos de ordenación *quicksort* y *mergesort*...
   (a) ...tienen el mismo coste temporal asintótico en el caso mejor.
   (b) ...tienen el mismo coste temporal asintótico en el caso peor.
   (c) ...tienen el mismo coste temporal asintótico tanto en el caso peor como en el caso mejor.

15. Indicad cuál de estas tres expresiones es falsa:

    (a) $\Theta(n/2) = \Theta(n)$
    (b) $\Theta(n) \subseteq O(n)$
    (c) $\Theta(n) \subseteq \Theta(n^2)$

16. Tenemos n substancias diferentes en polvo y queremos generar todas las distintas formas de mezclarlas de forma que el peso total no supere un gramo. Como la balanza que tenemos solo tiene una precisión de 0.1 gramos, no se considerarán pesos que no sean múltiplos de esta cantidad. Queremos hacer un programa que genere todas las combinaciones posibles.

    (a) No hay ningún problema en usar una técnica de vuelta atrás.
    (b) No se puede usar vuelta atrás porque las decisiones no son valores discretos.
    (c) No se puede usar vuelta atrás porque el número de combinaciones es infinito.

17. Qué tiene que valer b en la relación de recurrencia

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

    para que $T(n) = 2^n$?

    (a) 1
    (b) 2
    (c) 0

18. Queremos resolver por ramificación y poda el problema de la mochila discreta. Si resolvemos el mismo problema de forma voraz pero sin ordenar previamente los objetos por valor/peso, obtendremos:

    (a) Una cota pesimista.
    (b) Una cota optimista.
    (c) Nada que podamos utilizar.

19. Las relaciones de recurrencia...

    (a) ... aparecen sólo cuando la solución sea del tipo *divide y vencerás*.
    (b) ... expresan recursivamente el coste temporal de un algoritmo.
    (c) ... sirven para reducir el coste temporal de una solución cuando es prohibitivo.

20. El coste temporal asintótico del programa

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

    y el del programa

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

    son...

    (a) ... el del primero, menor que el del segundo.
    (b) ... el del segundo, menor que el del primero.
    (c) ... iguales.

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

    (a) El coste temporal promedio.

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

    (c) Nada de interés.

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

    (a) ... un algoritmo del estilo de *divide y vencerás*.

    (b) ... un algoritmo de programación dinámica.

    (c) ... un algoritmo voraz.

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

    (a) ... que ordenan el vector sin usar espacio adicional.

    (b) ... que se ejecutan en tiempo $O(n)$.

    (c) ... que aplican la estrategia de *divide y vencerás*.

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

    (a) Mediante un vector de booleanos.

    (b) Mediante un vector de reales.

    (c) Este problema no se puede resolver usando ramificación y poda si no se fija una cota superior al número total de aplicaciones de funciones.

25. Queremos resolver por vuelta atrás el problema de las n reinas. El usar una buena cota optimista permitiría:

    (a) No es aplicable ese tipo de podas a este problema.

    (b) Muy probablemente, hacer que el programa vaya mas lento.

    (c) Muy probablemente, resolver el problema de forma mas rápida.

26. Sea V el conjunto de todos los valores faciales que presentan las monedas de un país (por ejemplo, en la zona Euro, $1, 2, 5, 10, 20, 50, 100, 200\}$. Una cantidad M se puede entregar con varias combinaciones de monedas, pero hay una que usa el mínimo numero de monedas $n(M)$, el cual se puede calcular recursivamente como

$$n(M) = \min_{v \in V} (1 + n(M - v)),$$

    con las condiciones $n(0) = 0$ y, para $x < 0$, $n(x) = +\infty$. Indicad cuál de las siguientes afirmaciones es falsa:

    (a) El algoritmo que calcularía $n(M)$ así sería un algoritmo voraz y tendría un coste razonable.

    (b) El algoritmo recursivo que calcularía $n(M)$ así tendría en general un coste prohibitivo porque calcularía $n(x)$ muchas veces para la misma x.

    (c) El algoritmo recursivo que calcularía $n(M)$ se podría convertir en un algoritmo con un coste razonable usando memoización.

27. El arbol de expansión de mínimo coste de un grafo ...

(a) ... puede utilizarse como cota optimista para resolver el problema del viajante de comercio.
(b) ... puede utilizarse como cota pesimista para resolver el problema del viajante de comercio.
(c) Ninguna de las otras dos opciones es verdadera.

28. Pertenece $3n^2$ a $O(n^3)$?

(a) No.
(b) Sí, pero sólo si se toma $c > 3$
(c) Ninguna de las otras dos opciones son verdaderas.

29. 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.

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

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

31. Si $\lim_{n\to\infty}(g(n)/f(n))$ resulta ser una constante positiva no nula, cuál de las siguientes expresiones NO puede darse?

(a) $f(n) \in \Theta(g(n))$
(b) $g(n) \notin \Theta(f(n))$
(c) $f(n) \in \Omega(g(n))$ y $g(n) \in \Omega(f(n))$

32. Tenemos un vector ordenado de tamaño n_o y un vector desordenado de tamaño n_d y queremos obtener un vector ordenado con todos los elementos. ¿Qué será mas rápido?

(a) Ordenar el desordenado y luego mezclar las listas.
(b) Insertar los elementos del vector desordenado (uno a uno) en el vector ordenado.
(c) Depende de si $n_o > n_d$ o no.

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

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

34. Tenemos una lista ordenada de tamaño n_o y una lista desordenada de tamaño n_d, queremos obtener una lista ordenada con todos los elementos. ¿Cuál sería la complejidad de insertar, uno a uno, todos los elementos de la lista desordenada en la ordenada.

(a) $O(n_d \log n_o)$
(b) $O(n_o \times n_d + n_d^2)$
(c) $O(n_d \times n_o)$

35. Tenemos una función recursiva con la siguiente cabecera:
```
double f( const double &)
```
Con sólo esta información, cuál podría ser una definición adecuada para el almacén?

(a) Ninguna de las otras dos opciones son verdaderas.
(b) vector <double> A;
(c) int A[];

36. Queremos resolver por ramificación y poda el problema de la mochila discreta. Si resolvemos el mismo problema de forma voraz permitiendo coger objetos fraccionados pero sin ordenar previamente los objetos por valor/peso, obtendremos:

(a) Nada que podamos utilizar.
(b) Una cota pesimista.
(c) Una cota optimista.

37. ¿Cual es el coste espacial asintótico del siguiente algoritmo?
```
int f( int n ) {
  int a = 1, r = 0;
  for( int i = 0; i < n; i++ ) {
    r = a + r;
    a = 2*r;
  }
return r;
}
```

(a) $O(1)$
(b) $O(\log(n))$
(c) $O(n)$

38. Qué diferencia (entre otras) hay entre el algoritmo de Prim y el Kruskal?

(a) El subgrafo que paso a paso va generando el algoritmo de Prim siempre contiene una única componente conexa mientras que el de Kruskal no tiene por qué.
(b) El algoritmo de Prim es voraz y el de Kruskal no.
(c) Aún siendo el grafo de partida totalmente conexo, el algoritmo de Kruskal garantiza la solución óptima mientras que el de Prim sólo garantiza un subóptimo.

39. 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 $T(n) \in \Omega(n^2)$

40. 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.