Algoritmos: Evaluación Continua
curso 2019-2020

1. Dibuje los montículos que intervienen en la ordenación del siguiente vector con la Ordenación por montículos:

   5 2 0 3 7 1 3 2 4 6

   ¿Por qué la complejidad de este algoritmo es $O(nlogn)$?

2. Dibuje el árbol de recursividad que produce la Ordenación Rápida con Mediana de 3 a partir de la entrada siguiente, suponiendo un umbral = 4 y usando índices 1..n:

   14 7 9 6 2 1 13 8 3 12 5 10 4

   Presente el estado del vector antes de llamar a Ordenación por Inserción, y determine el número de inversiones por resolver.

3. Dibuje el árbol de recursividad que produce la Ordenación Rápida con pivote aleatorio a partir de la entrada siguiente para un mejor caso, suponiendo un umbral = 4:

   13 7 9 6 2 1 12 8 3 11 5 10 4

   Presente el estado del vector antes de llamar a Ordenación por Inserción, y determine el número de inversiones por resolver.

4. Escriba el estado del vector tras la ejecución de cada fase del algoritmo de Shell con incrementos de Hibbard (siendo 15 el primero) a partir de la entrada siguiente:

   10 18 9 7 4 2 1 15 13 16 8 3 12 5 14 17 6

   ¿Cuántos intercambios fueron necesarios para colocar el número **2** en su posición final? ¿Cuántos serían necesarios con la ordenación por Inserción?

5. Considerando un sistema monetario $M = \{v_1, v_2, ...v_m\}$, se dispone de una función Monedas que aplica la técnica de Programación Dinámica para encontrar el número mínimo de monedas para pagar la cantidad n, calculando para ello una tabla T con todos los resultados intermedios ($T[i, j]$: solución óptima para pagar la cantidad j con las monedas $v_1..v_i$). El resultado encontrado puede corresponder a una o varias configuraciones (conjuntos de monedas que suman n).

   Diseñe el pseudocódigo de una función Composición que, a partir de la tabla T ya construida por la función Monedas, devuelva una configuración óptima para el pago de la cantidad n, especificando el conjunto de monedas que lo componen. Previamente, proponga un tipo de datos adecuado para la salida de la función Composición.

6. Dada una mochila, de capacidad $W = 13$ unidades de peso, y el conjunto siguiente de *objetos no fraccionables*, caracterizados por su valor (v) y su peso (w):

objeto	1	2	3	4	5	6
v	3	4	6	5	5	6
w	2	3	4	5	6	7

   a) Construya la tabla con la que podría determinar en programación dinámica la carga más valiosa posible para esta mochila.

b) Dibuje sobre la tabla anterior tres recorridos que correspondan a tres soluciones, que identificará en su respuesta.

c) Justifique la solución que daría un algoritmo voraz que seleccione los objetos según su rentabilidad. Conclusión?

7. Programación Dinámica:

   a) Construya la tabla con la que podría determinarse en programación dinámica la manera óptima de pagar una cantidad de 12 unidades de valor con un mínimo de monedas, sabiendo que el sistema monetario considerado está constituido por monedas de 1, 4, 6, 8 y 10 unidades de valor.

   b) Indique la(s) solución(es) al problema dibujando su(s) correspondiente(s) traza(s) en la tabla anterior para justificar cómo la(s) obtiene.

   c) ¿Porqué descartaría el uso de la técnica voraz para resolver este problema?

8. Se dispone de n objetos no fraccionables $1,2,3,\ldots n$, caracterizados por su peso $w[i]$ y su valor $v[i]$. Una función *Mochila* permite encontrar, aplicando la técnica de *Programación Dinámica*, la carga de valor máximo (V) para una mochila limitada por el peso máximo que puede contener (W): para ello, calcula una tabla $M[1..n, 0..W]$ con todos los resultados intermedios $M[i, j]$ (i objetos, capacidad j). La solución $V = M[n, W]$ puede corresponder a una o varias configuraciones de la carga, constituidas por uno o varios conjuntos de objetos, sabiendo que no caben todos.

   Se plantea el diseño de una función *Composición* que, a partir de la tabla M construida por la función *Mochila*, devuelva una configuración posible de la solución, especificando el conjunto de objetos que la componen:

   a) Proponga un tipo de datos adecuado para la salida de la función *Composición*.

   b) Proponga un pseudocódigo de la función *Composición* que priorice los objetos más pesados, suponiendo que los objetos se han incorporado a la tabla ordenados por su peso (primero el más ligero).