Análisis y Diseño de Algoritmos

Ejercicios de Programación Dinámica

Francisco Javier Mercader Martínez

1. En el problema de la mochila (igual que en el problema del cambio de monedas) puede existir en general de una solución óptima para unas entradas determinadas. ¿Cómo se puede comprobar si una solución óptima es única o no, suponiendo que hemos resuelto el problema utilizando programación dinámica? Dar un algoritmo para que, a partir de las tablas resultantes del problema de la mochila, muestre todas las soluciones óptimas existentes.

2. El número de combinaciones de `m` objetos entre un conjunto de `n`, denotado por $\binom{n}{m}$, para $n \geq 1$ y $0 \leq m \leq n$, se puede definir recursivamente por:

   $$\binom{n}{m} = 1 \quad \text{Si } m = 0 \text{ ó } m = n$$
   $$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1} \quad \text{Si } 0 < m < n$$

   Conociendo que el resultado puede ser calculado también con la fórmula: $\dfrac{n!}{m!(m-n)!}$

   a. Dar una función recursiva para calcular $\binom{n}{m}$, usando la primera de las definiciones. ¿Cuál será el orden de complejidad de este algoritmo? Sugerencia: la respuesta es inmediata.

   b. Diseñar un algoritmo de programación dinámica para calcular $\binom{n}{m}$. Nota: la tabla construida por el algoritmo es conocida como el *"triángulo de Pascal"*. ¿Cuál será el tiempo de ejecución en este caso?

3. Considerar el problema de la mochila 0/1. En este ejercicio estamos interesados en calcular el número de formas distintas de meter o no los objetos en la mochila, pero respetando la capacidad máxima de la mochila. Por ejemplo, si todos los `n` objetos en la mochila, existirán 2^n formas posibles. Pero en general, si no caben todos, habrán muchas menos. Resolver mediante programación dinámica el problema de calcular el número de formas distintas de completar total o parcialmente la mochila. Datos del problema: `n` objetos, `M` capacidades de la mochila, $p = (p_1, p_2, \ldots, p_n)$ pesos de los objetos.

4. Una variante del problema de la mochila es la siguiente. Tenemos un conjunto de enteros (positivos) $A = \{a_1, a_2, a_3\}$ y un entero K. El objetivo es encontrar si existe algún subconjunto de A cuya suma sea exactamente K.

   a. Desarrollar un algoritmo para resolver este problema, utilizando programación dinámica. ¿Cuál es el orden de complejidad del algoritmo?

   b. Mostrar cómo se puede obtener el conjunto de objetos resultantes (en caso de existir solución) a partir de las tablas utilizadas por el algoritmo.

   c. Aplicar el algoritmo sobre el siguiente ejemplo $A = \{2, 3, 5, 2\}$, $K = 7$. ¿Cómo se puede comprobar que la solución no es única?

5. En el problema de la mochila 0/1 disponemos de dos mochilas, con capacidades **M1** y **M2**. El objetivo es maximizar la suma de beneficios de los objetos transportados en ambas mochilas, respetando las capacidades de cada una. Resolver el problema mediante programación dinámica, definiendo la ecuación recurrente, las tablas usadas y el algoritmo para rellenarlas.

   Datos del problema: **n** objetos, **M1** capacidad de la mochila 1, **M2** capacidad de la mochila 2, $p = (p_1, p_2, \dots, p_n)$ pesos de los objetos, $b = (b, b, \dots, b_n)$ beneficios de los objetos.

6. Una agencia de turismo realiza planificaciones de viajes aéreos. Para ir de una ciudad **A** a **B** puede ser necesario coger varios vuelos distintos. El tiempo de un vuelo directo de **I** a **J** será **T[I, J]** (que puede ser distinto de **T[J, I]**). Hay que tener en cuenta que si cogemos un vuelo (de **A** a **B**) y después otro (de **B** a **C**) será necesario esperar un tiempo de "escala" adicional en el aeropuerto (almacenado en **E[A, B, C]**).

   a. Diseñar una solución para resolver este problema utilizando programación dinámica. Explicar cómo, a partir de las tablas, se puede obtener el conjunto de vuelos necesarios para hacer un viaje concreto.

   b. Mostrar la ejecución del algoritmo sobre la siguiente matriz **T**, suponiendo que todos los **E[A, B, C]** tienen valor 1. ¿Cuál es el orden de complejidad del algoritmo?

T[i, j]	A	B	C	D
A	-	2	1	3
B	7	-	9	2
C	2	2	-	1
D	3	4	8	-

7.