Optimización - 2013 - Práctica 2

Programación Dinámica

1. Escribir un algoritmo para calcular los coeficientes binomiales usando el triángulo de Pascal. Determinar las complejidades temporal y espacial del algoritmo. Aclaración: para determinar la complejidad espacial tienen que calcular el tamaño de toda la información guardada hasta terminar el algoritmo, en función del tamaño de la entrada.

2. Sea una matriz de números naturales $M \in \mathbb{N}^{m \times n}$. Se desea obtener un camino que empiece en la casilla superior izquierda $(1, 1)$, termine en la casilla inferior derecha (m, n), y tal que minimice la suma de los valores de las casillas por las que pasa. En cada casilla (i, j) hay dos movimientos posibles: ir hacia abajo (a la casilla $(i + 1, j)$), o ir hacia la derecha (a la casilla $(i, j+1)$). Diseñe un algoritmo mediante programación dinámica que permita resolver este problema. Determine la complejidad.

3. Implementar un algoritmo usando la técnica de programación dinámica para poder resolver el problema de la mochila (versión clásica). Se tienen n ítems con pesos en kilos $a_1, \ldots, a_n$ y como máximo se pueden cargar M kilos. Cada ítem tiene un valor de utilidad v_i. Se desea encontrar una elección posible de los ítems tal que su peso total sea menor o igual a M y además su utilidad total sea máxima. Estudiar la complejidad. ¿Diría usted que es un algoritmo eficiente?

4. Agregamos al problema de la mochila los datos $v_1, ..., v_n$, que son los volúmenes enteros de cada ítem y la restriccón de que el volumen total no puede superar a un dado entero $V \geq 0$. Resolver con estas nuevas condiciones.

5. Una persona tiene un presupuesto de dinero $P \in \mathbb{N}$ para invertir en N objetos. Si invierte la cantidad entera $x_i \geq 0$ en el objeto i entonces recibirá una utilidad $u_i(x_i)$, que es una función no decreciente. Esta persona se pregunta cómo distribuir su presupuesto para las inversiones con el objetivo de optimizar la suma de las utilidades sobre todos los objetos. Resolver con Programación Dinámica.

6. Tenemos que producir acero en láminas de distinto ancho. La demanda de medidas para los anchos es $m_1, m_2, ..., m_N, m_{N+1}$ pero nos es imposible disponer en stock de todos los anchos pedidos. Sólo podemos guardar láminas de ancho m_{N+1} y de $n < N$ anchos distintos más. Cuando se quiere cubrir el pedido de un ancho m_j del que no se dispone, se toma una lámina del menor ancho $m_i > m_j$ y se le cortan los bordes, suprimiendo el exceso $m_i - m_j$, lo que produce un costo de descarte $d(i, j)$. El problema es, dado un entero n, y si disponemos del ancho máximo m_{N+1}, cómo debemos elegir los n anchos de entre los primeros N de manera que el descarte total sea mínimo.

7. Se tiene un grafo dirigido acíclico $G = (V, E)$ con costos no negativos en las aristas. Sea V_0 el conjunto de vértices terminales, donde decimos que un vértice v es terminal si no existe ningún vértice w tal que $v \to w \in E$. Diseñar un algoritmo para calcular el camino de menor costo total, partiendo de un vértice $u \in V$ y llegando a algún vértice de V_0, usando Programación Dinámica. Determinar la complejidad.
   Si bien la Programación Dinámica se puede aplicar con costos de cualquier signo, en este ejercicio se imponen costos no negativos para comparar la solución con la que resulta del algoritmo de Dijkstra. Compare la complejidad de su solución con la complejidad de este último algoritmo.

8. Siguiendo en el esquema anterior, supongamos nuevamente G un grafo dirigido y acíclico, representando una red vial en una región montañosa. Cada vértice representa una ciudad y cada arista un trecho de camino, de cada trecho se registra la altura máxima del mismo. La idea es determinar un camino desde la ciudad A hasta alguna de las ciudades terminales (V_0) de manera tal de minimizar la máxima altura por la cual debe transitar.

9. Se quiere sumar un importe M usando el mínimo número posible de monedas.

   a) Implementar un algoritmo para poder cumplir el requerimiento si se dispone de monedas de 1,5, 10 y 25 centavos. Suponga que se tiene una cantidad infinita de monedas de cada denominación. Calcule la complejidad.

   b) Escriba ahora un algoritmo para el mismo problema con n denominaciones arbitrarias $d_1, \ldots, d_n$.