Ejercicios en clase: Prog. Dinámica (lista en crecimiento)

Análisis y Diseño de Algoritmos

23 de junio de 2023

Ejercicio 1. Estudie el problema Rod cutting de la sección 15.1 del libro Cormen. Luego de dicho estudio, ejecute el algoritmo BOTTOM-UP-CUT-ROD con la siguiente entrada: $p = [1; 3; 4; 5; 9; 9; 11; 12; 15]$. Muestre en cada paso como se llena cada posición del arreglo resultante. Finalmente, indique cual es la secuencia de cortes que corresponde a la solución óptima obtenida por el algoritmo.

Ejercicio 2. Estudie el algoritmo MATRIX-CHAIN-ORDER (Cormen, sección 15.2). Luego de dicho estudio, ejecute el algoritmo con la siguiente entrada: [10; 20; 6; 24; 10; 100; 12]. Muestre la tabla resultante de aplicar el algoritmo, así como la parentización óptima obtenida luego de aplicarlo.

Ejercicio 3. Muestre la tabla para el problema SUBSET-SUM visto en clase para los items $[2, 3, 2, 4, 8, 9, 5, 7]$ con $W = 10$. Solo es necesario que muestre la tabla resultante final.

Ejercicio 4. Muestre la tabla para el problema MIN-PARTITION visto en clase para el arreglo $[23, 2, 1, 2, 18, 22, 14, 21, 6, 5]$ con $k = 5$. Solo es necesario que muestre la tabla resultante final.

Ejercicio 5. Diseñe un algoritmo de programacion dinámica para contar el número de maneras en que podemos construir una suma n, dada como entrada, lanzando un dado una o más veces. Por ejemplo, si $n = 3$. existen cuatro maneras $(1 + 1 + 1, 1 + 2, 2 + 1, 3)$. Debe indicar la recurrencia que resuelve el problema y el tiempo de ejecución de su algoritmo.

Ejercicio 6. Usted tiene n tipos de monedas a disposición: $c_1, c_2, \ldots, c_n$. Además desea cambiar un monto x de manera tal que el número de monedas usadas sea el mínimo posible. Por ejemplo, si $x = 11$ y tiene las monedas $1, 5, 7$ entonces puede cambiar dicha suma usando tres monedas: $11 = 1 + 5 + 5$. Indique cual es la recurrencia que resuelve el problema y demuestre que dicha recurrencia está correcta. A partir de ello, diseñe un algoritmo de programación dinámica con complejidad $O(nx)$ para el problema anterior.

Ejercicio 7. Considere el problema de subarreglo de suma máxima: dado un arreglo $A[1..n]$ de números enteros, encontrar un subarreglo con la mayor suma posible.

Sea $OPT(i)$ el valor de la suma máxima en un subarreglo del arreglo $A[1..n]$ **que termina en** i.

- Encuentre una recurrencia para $OPT(i)$
- Demuestre que su recurrencia está correcta usando la propiedad de subestructura óptima.
- Diseñe un algoritmo de programación dinámica $O(n)$ apartir de los items anteriores.

Ejercicio 8. Considere el siguiente problema. Usted recibe dos secuencias $a_1, a_2, \ldots, a_n$ y $b_1, b_2, \ldots, b_n$. Desea encontrar una secuencia de índices $i_1 < i_2 < \ldots < i_k$ tal que k es máximo y se cumple que $a_{i_1} < a_{i_2} < \cdots a_{i_k}$ pero $b_{i_1} > b_{i_2} > \cdots > b_{i_k}$. Indique cual es la recurrencia que resuelve el problema y demuestre que dicha recurrencia está correcta. A partir de ello, diseñe un algoritmo de programación dinámica con complejidad $O(n^2)$ para el problema anterior.

Ejercicio 9. Debes cortar un tronco en varias piezas. La empresa la mejor forma de hacerlo es con Analog Cutting Machinery (ACM), que cobra según el longitud de registro a cortar. Su máquina de corte permite que solo se haga un corte a la vez. Si queremos hacer varios cortes, es fácil ver que diferentes órdenes de estos cortes conducen a precios muchos diferentes. Por ejemplo, considere un registro de 10 metros de largo, que debe ser cortado a 2, 4 y 7 metros de uno de sus extremos. Hay varias posibilidades. Podemos primero cortar a 2 metros, luego a 4 y después a 7. Este pedido cuesta $10 + 8 + 6 = 24$, porque el primer tronco tenía 10 metros de largo, lo que quedaba era de 8 metros de largo y el la última pieza era de longitud 6. Si cortamos en orden 4, luego 2, luego 7, pagaríamos $10 + 4 + 6 = 20$, que es más barato.

Diseñar un algoritmo de programación dinámica que, dada la longitud ℓ del tronco y k puntos $p_1, \ldots, p_k$ de corte, encuentre el costo mínimo para realizar estos cortes. Indique cual es la recurrencia usada.

Ejercicio 10. Considere un tablero $n \times n$ de ceros y unos. Un 1 indica que puedo pisar en dicha casilla del tablero y un 0 indica que no es posible pisar dicha casilla.

Una secuencia de casillas es considerada un camino si puedo pisar dichas casillas y las cada par de casillas adyacentes en la secuencia tienen un lado en común. Además, consideremos que solo podemos movernos o bien la derecha, o bien hacia abajo.

El problema consiste en calcular el número de caminos desde la casilla $(1, 1)$ hasta la casilla (n, n). Escriba una recurrencia que permita resolver el problema. Pruebe la correctitud de su recurrencia. Diseñe un algoritmo de programación dinámica a partir de la recurrencia.

Ejercicio 11. Los transbordadores se utilizan para transportar automóviles entre una orilla de un río y la otra. Funcionan acomodando dos carriles de automóviles a lo largo de toda su longitud. Los autos entran a los carriles por un lado del transbordador y salen, en la otra orilla, al otro lado del transbordador.

La fila de espera de automóviles para ingresar al transbordador es una sola línea y hay un operador dirige cada automóvil a cualquiera de las dos pistas del transbordador, el carril izquierdo o el carril derecho, a manera de equilibrar las dos pistas. Cada automóvil tiene una longitud diferente y un operador decide cuál de los dos carriles cada automóvil debe abordar. El objetivo es abordar tantos automóviles como sea posible (note que no es posible saltearme un automovil de la fila). Diseñar un algoritmo que determina el maximo número de carros que puedo cargar en el transbordador.

Más precisamente, sea $A[1..n]$ un arreglo con longitudes de n automoviles, y sea ℓ la longitud del transbordador. Considere que la lista de espera de los automóviles es $A[1], A[2], \ldots, A[n]$. Es decir, primero procesaremos el automovil con longitud $A[1]$, luego el automovil con longitud $A[2]$, y así sucesivamente. Queremos determinar cual es el mayor número k tal que los automóviles con longitudes $A[1], \ldots, A[k]$ pueden ser cargados en la balsa. Diseñar un algoritmo de programación dinámica para este problema e indique cual es la recurrencia usada.

Ejercicio 12. Suponga que tenemos una secuencia $\langle x_1, \ldots, x_{2n+1} \rangle$ en la cual $\langle x_1, x_3, \ldots, x_{2n+1} \rangle$ son enteros positivos y $\langle x_2, x_4, \ldots, x_{2n} \rangle$ pertenecen al conjunto $\{+, *\}$. Queremos insertar paréntesis en la secuencia de tal forma que el valor de la expresión aritmética resultante sea máximo. Por ejemplo, si la secuencia dada es 2+4*1+5 entonces una solución es la expresión $((2 + 4) * (1 + 5))$, que vale 36.

- Encuentre una recurrencia que devuelve el valor máximo obtenible luego de aplicar paréntesis a una secuencia. La recurrencia debe permitir resolver el problema en $O(n^3)$
- Demuestre que su recurrencia es correcta utilizando argumentos de subestructura óptima.
- Utilizando la recurrencia, escriba el pseudocódigo de un algoritmo de programación dinámica que permita resolver el problema.

Ejercicio 13. Supongamos que tienes un máquina y un conjunto de n trabajos, identificados por los números $1, 2, ..., n$, para procesar en esa máquina. Cada trabajo j tiene un tiempo de procesamiento t_j, un beneficio p_j y una fecha límite fin d_j. La máquina solo puede procesar un trabajo a la vez, y el trabajo debe realizarse ininterrumpidamente durante t_j unidades de tiempo consecutivas. Si el trabajo ya está completado en su fecha límite d_j, usted recibe una ganancia p_j, pero si se completa después de su fecha límite, no recibe ganancias. Diseñe un algoritmo de programación dinámica que encuentra el orden de ejecución de los trabajos que maximiza la suma de beneficios, suponiendo que todos los tiempos de procesamiento son enteros entre 1 y n. Indique cual es la recurrencia usada ¿Cuál es el tiempo de ejecución de su algoritmo?