UNIVERSIDAD DE MÁLAGA

Análisis y Diseño de Algoritmos
Programación Dinámica
(2º de Grado Ing. Inf., Ing. Sw., Ing. Comp.)
E.T.S.I. INFORMÁTICA

Nota: Al final de la relación hay algunas pistas para la resolución de ciertos apartados (los que están marcados con el símbolo ★). Se recomienda recurrir a dichas pistas sólo tras haber hecho un intento de resolver los correspondientes ejercicios sin las mismas.

Ejercicios Básicos

1. Sean $A_1, \cdots, A_n$ un conjunto de matrices que se desean multiplicar en secuencia. Se supone que estas matrices son compatibles, esto es, la dimensión de A_i es $m_i \times n_i$ y se cumple que $n_i = m_{i+1}$. Nótese que al multiplicar dos matrices de tamaños $m \times n$ y $n \times p$ se realizan $m \cdot n \cdot p$ multiplicaciones. El número total de multiplicaciones escalares que se realizan dependerá del modo en que se asocien las matrices para la multiplicación. Por ejemplo, sea $A_1^{10\times100}$, $A_2^{100\times5}$ y $A_3^{5\times50}$:

   - $((A_1A_2)A_3)$ supone $10 \cdot 100 \cdot 5 + 10 \cdot 5 \cdot 50 = 7500$ multiplicaciones escalares.
   - $(A_1(A_2A_3))$ supone $100 \cdot 5 \cdot 50 + 10 \cdot 100 \cdot 50 = 75000$ multiplicaciones escalares.

   Resulta por lo tanto fundamental encontrar la asociación óptima para realizar la multiplicación.

   a) Encontrar una expresión recurrente que indique cuántas parentizaciones posibles hay.

   b) ★ Analizar la estructura del problema y encontrar una forma recursiva de expresar dicha estructura en función de subproblemas más simples.

   c) Implementar un algoritmo de programación dinámica que calcule el número óptimo de multiplicaciones escalares necesarias para multiplicar una serie de matrices $A_1, \cdots, A_n$.

   d) Implementar una versión memoizada del algoritmo recursivo simple que emana de la recurrencia definida en $1b$.

2. Se desea construir una antena de telefonía de M metros de altura. Para ello se dispone de un suministro ilimitado de bloques de armazón de n tipos distintos, cada uno de altura a_i y peso p_i. El objetivo es determinar cuántos bloques hay que emplear de cada tipo, de manera que la antena mida exactamente M metros y tenga el peso mínimo.

   a) Demostrar que el problema exhibe la propiedad de subestructura óptima.

   b) Encontrar una expresión recurrente para el peso óptimo de la antena.

   c) Implementar un algoritmo de programación dinámica *bottom-up* que calcule el peso final óptimo de la antena.

   d) Ampliar el algoritmo anterior para que también devuelva el número óptimo de bloques que hay que usar de cada tipo.

3. Un estudiante aventajado intenta organizar sus horas de estudio para maximizar la nota acumulada que obtendrá en las n asignaturas del cuatrimestre. Para ello, analiza los temarios y obtiene una estimación c_{ij} de la nota numérica que obtendría en la asignatura i si le dedicara j de las H horas de estudio de las que dispone. Su objetivo ahora es determinar cuántas horas debe dedicar a cada asignatura.

   a) Demostrar que el problema exhibe la propiedad de subestructura óptima.

   b) Encontrar una expresión recurrente para la calificación global óptima.

   c) Implementar un algoritmo de programación dinámica *bottom-up* que calcule dicha calificación óptima, así como el número de horas que hay que dedicar a cada asignatura.

   d) Calcular la complejidad temporal de dicho algoritmo.

4. Dispones de un conjunto A de n enteros, y sea s un número entero. Se requiere calcular cuántos subconjuntos no vacíos de A suman exactamente s. Para ello se pide:

   a) Proporcionar la ecuación de Bellman que nos devuelva el número de subconjuntos que cumplen lo indicado.

   b) Implementar un algoritmo de programación dinámica *bottom-up* que siga la ecuación previamente indicada.

   c) Resolver la instancia del problema dada por los valores $A = \langle -1, 2, 3, -3, 5\rangle$, $s = 2$.

5. A lo largo de un río en el que sólo se puede navegar en una dirección se encuentran n embarcaderos. Conocemos el coste T_{ij} para ir directamente del embarcadero i al embarcadero j (y que puede ser más barato o más caro que un recorrido entre i y j con paradas intermedias). Se desea encontrar el coste mínimo para viajar entre cualesquiera embarcaderos i y j.

6. Como oficial de comunicaciones del crucero de combate klingon *IKS B'Moth*, su labor es la de gestionar las comunicaciones de la manera más eficiente. Supóngase que se quiere transmitir un mensaje $S = s_1 \cdots s_m$ dado como una cadena de m símbolos. A tal fin, dispone de r códigos distintos. Sea b_{ij} el número de bits necesarios para codificar el i-ésimo símbolo en el j-ésimo código. Inicialmente el trasmisor del puente está fijado al código #1 pero puede ser cambiado durante la transmisión cuando se desee. Para ello se necesita enviar un código de control compuesto de C_{ij} bits si se desa cambiar del código i actual al código j. Su objetivo es determinar cómo enviar el mensaje empleando el número mínimo de bits. *Qapla'!*

   a) Demostrar que el problema exhibe la propiedad de subestructura óptima

   b) Expresar mediante una ecuación de Bellman el número óptimo de bits.

   c) Construir un algoritmo de programación dinámica *bottom-up* en función de la anterior ecuación.

7. Dado un conjunto de n objetos, cada uno con un peso $P = \{p_1, p_2, \ldots, p_n\}$, supongamos que queremos repartir los objetos en dos montones diferentes, de manera que queden lo más equilibrados posible en peso. Para *conocer el desequilibrio mínimo* queremos aplicar un algoritmo de Programación Dinámica, y para ello se pide:

   a) Plantear una solución recursiva del mismo, aplicando el principio de óptimo.

   b) Definir una estructura de datos adecuada para resolver el problema de forma eficiente.

   c) Definir la composición y estructura de una segunda segunda tabla que permita conocer no solo el valor de la solución óptima, sino cómo se alcanza.

   d) Aplicar el algoritmo sobre el ejemplo $n = 4$ y $P = \{2, 1, 3, 4\}$, construir las correspondientes tablas y dar tanto la solución óptima como los dos subconjuntos de P para los que se obtiene.

8. Queremos jugar a un juego en el que tenemos un tablero nxn donde en cada casilla aparece un número natural. El juego consiste en elegir una casilla de la última fila (n) y movernos desde ella hasta una casilla de la primera fila (1), donde los únicos movimientos legales consisten en moverse, sin salirse del tablero, de una casilla a una de las tres casillas de la fila superior alcanzables en vertical ($\uparrow$) o en diagonal ($\nwarrow$ ó $\nearrow$). La cantidad ganada será la suma de los valores en las casillas del tablero por las que pasamos en este recorrido. Por ejemplo, si el tablero es el siguiente:

1	4	8	3
1	5	2	10
8	2	7	9
3	**5**	2	1

   Un posible recorrido que empiece en la casilla $(4, 2)$, y después pasar por las casillas $(3, 1), (2, 1)$ y $(1, 1)$ obtiene una ganancia de 15.

   a) Define la función de recurrencia para un tablero de nxn.

b) Resuelve el problema para la instancia del problema dado.

c) Escribir un algoritmo que determine, dado un tablero, cual es la casilla de la última fila por la que interesa empezar.

9. Dada una secuencia $L_1, \cdots, L_n$ de n listas **ordenadas** con $l_1, \cdots, l_n$ elementos, respectivamente, se pretende mezclarlas **por pares** hasta obtener una única lista ordenada realizando **el número mínimo de movimientos de datos**. Teniendo en cuenta que para mezclar de forma ordenada dos listas ordenadas L y L' de longitudes l y l' hay que hacer $l + l'$ movimientos de datos, la forma en la que se **asocian** las listas para mezclarse determina la eficiencia del proceso.

   Por ejemplo, supón que hay tres listas L_1, L_2 y L_3 con 30, 20 y 10 elementos respectivamente. Suponiendo que la mezcla ordenada de dos listas L, L' se representa como (LL'), la mezcla de las listas L_1, L_2 y L_3 puede puede realizarse de dos formas distintas:

   a) $((L_1L_2)L_3)$, es decir, primero mezclamos L_1 con L_2 (50 movimientos) y, a continuación, mezclamos el resultado con L_3 (60 movimientos), lo que da un total de 110 movimientos

   b) $(L1(L_2L_3))$, es decir, primero mezclamos L_2 con L_3 (30 movimientos) y, a continuación, mezclamos L_1 con el resultado (60 movimientos), lo que da un total de 90 movimientos

   Nota: Observa que en el proceso de mezcla, el orden de las listas no se modifica. Lo relevante es cómo se colocan los paréntesis.

   Se pide,

   a) Definir una recurrencia que exprese el número de formas distintas de ordenar las listas.

   b) Implementa un algoritmo de programación dinámica que resuelva la recurrencia del apartado anterior.

Ejercicios Complementarios

1. En este juego, que llamaremos el juego de *monedas en línea*, un número par, n, de monedas, de varias denominaciones, se colocan en fila. Dos jugadores, que vamos a llamar Alice y Bob, se turnan para quitar una de las monedas de cualquiera de los extremos de la línea restante de monedas. Es decir, cuando es el turno de un jugador, él o ella retira la moneda en el extremo izquierdo o derecho de la línea de monedas y agrega esa moneda a su colección. El jugador que elimina un conjunto de monedas con mayor valor total que el otro jugador gana, donde asumimos que tanto Alice como Bob conocen el valor de cada moneda, como, por ejemplo, si usamos dólares (Ver Figura 1).

Figura 1: Una instancia posible del problema

   a) ★ Proporcionar la ecuación de recurrencia para el máximo número de monedas cogidas por Alice si Bob juega de manera óptima.

   b) Implementar el algoritmo y detallar las estructuras usadas.

   c) Resolver la instancia del problema de la figura 1.

   d) ¿Puede resolverse este problema (obtener el máximo de monedas que Alice puede coger) mediante un algoritmo ávido? En caso afirmativo, proporcionar la estrategia (es decir, el heurístico que se implementaría en el algoritmo ávido). En otro caso, proporcionar un contra-ejemplo.

2. Sea $G(V, E)$ un grafo dirigido con n vértices, y sea $A = \{a_{ij}\} \in \mathbb{B}^{n \times n}$ su matriz de adyacencia, i.e., $a_{ij} =$`CIERTO` si, y sólo si, el arco $v_i \to v_j$ está en el grafo. Dada esta última matriz, se desea calcular su cierre transitivo, esto es, una matriz $T = \{t_{ij}\} \in \mathbb{B}^{n \times n}$ en la que $t_{ij} =$`CIERTO` si, y sólo si, hay un camino que va desde el nodo v_i al nodo v_j.

   a) ★ Analizar la estructura del problema y encontrar una forma recursiva de expresar dicha estructura en función de subproblemas más simples.

   b) Confeccionar un algoritmo de programación dinámica que dada A calcule el cierre transitivo T. ¿Puede optimizarse el empleo de memoria?

3. Dos equipos $\mathcal{A}$ y $\mathcal{B}$ juegan por el campeonato mundial de balonvolea al mejor de $2n - 1$ partidos (es decir, el primero que gane n partidos se hace con el campeonato). Se sabe que cada partido acaba en victoria de $\mathcal{A}$ con probabilidad p y en victoria de $\mathcal{B}$ con probabilidad $q = 1 - p$, y se desea conocer la probabilidad de que $\mathcal{A}$ gane el campeonato.

   a) Sea $P(i, j)$ la probabilidad de que $\mathcal{A}$ gane el campeonato dado que le quedan i partidos por ganar, y que a $\mathcal{B}$ le restan j victorias. Expresar $P(i, j)$ de manera recurrente.

   b) Implementar un algoritmo recursivo simple basado en la anterior recurrencia. Indicar cuál sería la llamada inicial.

   c) ★ Estimar una cota superior a la complejidad del algoritmo anterior.

   d) Defina un algoritmo de programación dinámica basado en la recurrencia encontrada en $3a$. ¿Cuál es su complejidad?

   e) Defina una versión memoizada del algoritmo recursivo encontrado en $3b$

4. Se dice que una cadena s es una subsecuencia de otra cadena w si todos los caracteres de s aparecen en w en el mismo orden, aunque no sea de manera consecutiva. Por ejemplo, "momia" es una subsecuencia de "mesopotamia" (i.e., "**m**es**o**pota**mia**"). Dadas dos cadenas w_1 y w_2 el problema de la subsecuencia más larga es hallar la cadena s más larga que es a la vez subsecuencia de w_1 y w_2.

   a) ★ Definir recursivamente la estructura del problema.

   b) Implementar una versión memoizada de un algoritmo recursivo basada en la anterior recurrencia.

   c) Implementar un algoritmo de programación dinámica *bottom-up* basada en la estructura encontrada.

5. Un desconocido (que dice ser familiar suyo, aunque no se parece en nada a Vd.) se presenta ante Vd. montado en un DeLorean y le da un anuario con todos los resultados de la quiniela del año próximo. Sabemos también el premio r_i que habrá en la semana i ($1 \leqslant i \leqslant n$) si jugamos la quiniela ganadora en dicha semana. Para no levantar sospechas, decidimos que no podremos ganar más de una vez cada $K > 0$ semanas (en otras palabras, deben transcurrir al menos $K - 1$ semanas entre dos premios consecutivos). Determinar cómo hay que jugar para maximizar la ganancia.

   a) Demostrar que el problema exhibe la propiedad de subestructura óptima.

   b) Encontrar una expresión recurrente para el premio óptimo.

   c) Confeccionar un algoritmo de programación dinámica bottom-up para resolver el problema.

   d) Aplicar a mano este algoritmo para resolver la instancia determinada por $K = 4$, $r_i \in \langle 4, 10, 3, 5, 9, 6, 4, 4, 7, 8 \rangle$. Se desea conocer el beneficio total óptimo, así como la solución que lo produce.

6. Se dice que una cadena s es una supersecuencia de otra cadena w si todos los caracteres de w aparecen en s en el mismo orden, aunque no sea de manera consecutiva. Por ejemplo, "camisa" es una supersecuencia de "casa" (i.e., "**ca**mi**sa**"). Dadas dos cadenas w_1 y w_2 el problema de la supersecuencia más corta es hallar la cadena s más corta que es a la vez supersecuencia de w_1 y w_2. Nótese que dicha cadena no tiene por qué ser única.

a) Demostrar que el problema exhibe la propiedad de subestructura óptima.

b) Encontrar una expresión recurrente para la longitud de la supersecuencia.

c) Confeccionar un algoritmo de programación dinámica bottom-up para resolver el problema.

d) Aplicar a mano este algoritmo para encontrar una supersecuencia de longitud mínima para "gato" y "ratón" (los acentos pueden ignorarse).

7. El afamado delantero Cipriano Romualdo ha dado con la clave para marcar el mayor número posible de goles: dosificar adecuadamente su esfuerzo. Tras un sofisticado estudio biomecánico, el preparador físico de su equipo ha determinado que después de descansar durante i partidos su tasa de acierto en el j-ésimo partido posterior es τ_{ij} (e.g., $\tau_{3,5}$ sería la tasa de acierto en el quinto partido que juega después de haber descansado durante 3 partidos). El objetivo del entrenador es optimizar la tasa acumulada de acierto (la suma de la tasa de acierto para los N partidos de la temporada, todos igual de importantes).

   a) Demostrar que el problema exhibe la propiedad de subestructura óptima.

   b) Encontrar una expresión recurrente para la tasa óptima.

   c) Confeccionar un algoritmo de programación dinámica *top-down* para este problema.

   d) Determinar la complejidad en espacio y en tiempo del algoritmo anterior.

8. Sean u y v dos cadenas de caracteres. Se desea transformar u en v con el mínimo número de operaciones elementales del tipo siguiente: eliminar un carácter, añadir un carácter, y cambiar un carácter. Por ejemplo, se puede pasar de `abbac` a `abcdc` en tres etapas:

   1) `abbac` $\longrightarrow$ `abac` (eliminamos `b` en la posición 3)
   2) $\longrightarrow$ `ababc` (añadimos `b` en la posición 4)
   3) $\longrightarrow$ `abcbc` (cambiamos `a` en la posición 3 por `c`)

   a) Mostrar que esta transformación no es óptima.

   b) Escribir un algoritmo de programación dinámica que calcule el número mínimo de operaciones necesarias para transformar u en v y cuáles son esas operaciones. Estudiar la complejidad del algoritmo en función de las longitudes de u y v.

9. Vd. es el director de logística de una empresa de componentes informáticos, y su trabajo es determinar al final de cada semana cómo se deben hacer llegar los componentes manufacturados de la fábrica al distribuidor. Para ello tiene dos opciones con su compañía de transportes: pagar en función del peso transportado, cargando K euros/kg, o pagar una tarifa plana de T euros semanales con independencia del peso. En este segundo caso, el contrato de tarifa plana debe hacerse durante s semanas consecutivas. Sabiendo que la producción de la fábrica es de p_i kg durante la semana i, $1 \leqslant i \leqslant n$, el objetivo es encontrar la forma menos costosa de distribuir los componentes.

   a) Demostrar que el problema exhibe la propiedad de subestructura óptima.

   b) Encontrar una expresión recurrente para el coste del transporte.

   c) Confeccionar un algoritmo de programación dinámica bottom-up para resolver el problema.

   d) Aplicar a mano este algoritmo para resolver la instancia determinada por $K = 1$, $T = 5$, $s = 3$, y $p_i \in \langle 4, 10, 3, 5, 1, 6, 4, 4\rangle$. Se desea conocer el coste total óptimo, así como la solución que lo produce.

10. Tienes la misión de controlar la calefacción central de una estación antártica. Para ello has recopilado infomración sobre las preferencias de tus compañeros en cada momento del día: sea c_{ij} el confort global de los miembros de la base si a la hora i ($1 \leqslant i \leqslant H$) la temperatura es de j grados. Al comienzo del día la temperatura se fija automáticamente a T grados y a partir de ahí puedes modificarla en como máximo K grados en cada paso/hora (pero nunca puedes salirte del rango $[T_{\text{mín}}, T_{\text{máx}}]$. Debes determinar qué temperatura tener en cada momento de manera que se maximice la satisfacción de toda la base.

    a) Demostrar que el problema exhibe la propiedad de subestructura óptima.

b) Encontrar una recurrencia para el confort óptimo.

c) Construir un algoritmo de programación dinámica para resolver el problema y determinar su complejidad.

11. Dada una tabla de tamaño $n \times n$ de números naturales, se pretende resolver el problema de obtener un camino de la casilla $(1, 1)$ a la casilla (n, n) que minimice la suma de los valores de las casillas por las que pasa. En cada casilla (i, j) habrá sólo dos posibles movimientos: ir hacia abajo $(i+1, j)$, o hacia la derecha $(i, j + 1)$. Aplicando Programación Dinámica, deseamos resolver el problema, para ello:

    a) Plantear la relación de recurrencia y las condiciones iniciales necesarias.

    b) Implementar el algoritmo.

    c) Construir la tabla de costes y la información necesaria para reconstruir la solución sobre la siguiente entrada:

2	8	3	4
5	3	4	5
1	2	2	1
3	4	6	5

12. Suponemos una red de carreteras que une distintas ciudades. Dadas dos ciudades puede ocurrir que no exista carretera directa sino que sea necesario pasar por otras ciudades. Para cada par de ciudades con conexión directa conocemos la categoría de la carretera que las une. Si una ruta esta formada por varios tramos, la categoría de la conexión será el mínimo entre las categorías de los tramos que la forman. Dadas dos ciudades, queremos saber cuál es la ruta de categoría máxima que las une. Disponemos de una matriz *categoría* que define la categoría de cada conexión directa entre dos ciudades (si no existe conexión, el correspondiente valor en la matriz será $-\infty$). Para resolver el problema queremos aplicar un algoritmo de Programación Dinámica, y para ello se pide:

    - Demostrar que el problema exhibe la propiedad de subestructura óptima
    - Expresar mediante una ecuación de Bellman la categoría óptima del camino entre cualesquiera dos ciudades.
    - Construir un algoritmo de programación dinámica *bottom-up* en función de la anterior ecuación.

Pistas e Indicaciones

Ejercicios Básicos

1. *b)* Sea M_{ij} el número óptimo de multiplicaciones escalares necesarias para multiplicar $A_i A_{i+1} \cdots A_j$. Encontrar una expresión recurrente para M_{ij}.

Ejercicios Complementarios

1. *a)* Considerar subproblemas del tipo $T^{(k)}$, en los que $t_{ij}^{(k)}$ =`CIERTO` si, y sólo si, hay un camino que va desde el nodo v_i al nodo v_j cuyos nodos intermedios (si los hay) sólo pueden ser $v_1, \cdots, v_k$. Encontrar una expresión recurrente para $t_{ij}^{(k)}$.

2. *c)* Exprese la complejidad de $P(i, j)$ como $t(w)$, donde $w = i + j$.

3. *a)* Sea n_i la longitud de w_i. Definir la estructura del problema en función de qué ocurre con la subsecuencia óptima s cuando $w_1[n_1] = w_2[n_2]$ y cuando $w_1[n_1] \neq w_2[n_2]$

4. *a)* Tenga en cuenta que el hecho de que Bob juegue de manera óptima, obliga a asumir en la ecuación que minimizará las oportunidades de Alice dentro de sus posibilidades/decisiones.