Métodos Algorítmicos en Resolución de Problemas

Grado en Ingeniería Informática

Hoja de ejercicios 6 Curso 2018-2019

Ejercicios de programación dinámica

Ejercicio 1 Tenemos n varillas distintas de longitudes enteras $l_1, \ldots, l_n$ y precios reales $c_1, \ldots, c_n$, respectivamente. Las varillas no se pueden cortar. Se desea soldar algunas de ellas para obtener una varilla de longitud total L (también entera).

Se pide escribir algoritmos de programación dinámica para resolver directamente cada uno de los siguientes problemas:

1. Indicar si es posible o no obtener la varilla deseada soldando algunas de las varillas dadas.
2. Calcular el número total de maneras de obtener la varilla deseada soldando algunas de las varillas dadas, sin que importe el orden de soldadura.
3. Minimizar el número de varillas utilizadas.
4. Minimizar el precio total de las varillas utilizadas.

Cada algoritmo tiene que resolver exactamente el problema pedido en cada caso y no hay que calcular en ningún caso las varillas necesarias. Por supuesto, hay que minimizar los costes en espacio y tiempo.

Ejercicio 2 Dado un vector $V[1..n]$ de números enteros mayores que 0, diseña un algoritmo que encuentre un subconjunto de suma $C > 0$ de cardinal mínimo.

Ejercicio 3 Debido a la gran afluencia de público en pequeñas barcas a la Copa del América, las autoridades portuarias han habilitado un embarcadero con anchura suficiente para que atraquen dos filas de embarcaciones, de longitud L cada una. Para amarrar, las embarcaciones esperan a la entrada del embarcadero en una única fila y una lancha del servicio de guardacostas se encarga de dirigirlas a la fila de la izquierda o la de la derecha. Cada embarcación tiene una eslora determinada e_i, de tal forma que $\sum_{i=1}^{N} e_i \geq 2L$, siendo N el número de embarcaciones en la cola. Diseñar un algoritmo que dada una cola de N embarcaciones ayude a los guardacostas a maximizar el número de estas que pueden amarrar en el embarcadero. Analizar el coste en tiempo y en espacio del algoritmo propuesto.

Ejercicio 4 (**El rapero con sueltecillo**) Un rapero lleva excesivo dinero suelto en el bolsillo de su pantalón, lo que contribuye a que lleve los pantalones demasiado caídos. Como consecuencia, cada vez que paga una cantidad C, su deseo es emplear el mayor número posible de monedas. Suponiendo que para todo i, $1 \leq i \leq n$, tiene n_i monedas del tipo m_i, diseñar un algoritmo que le diga cuáles monedas debe emplear.

Ejercicio 5 Inspirarse en el algoritmo de Floyd para diseñar un algoritmo que calcule el cierre reflexivo y transitivo de una relación binaria dada mediante un grafo dirigido.

Ejercicio 6 Sobre un río hay N embarcaderos en cada uno de los cuales se puede alquilar un bote para ir a cualquier otro embarcadero río abajo (es casi imposible remontar la corriente). La tarifa indica el coste del viaje desde i hasta j para cualquier punto de partida i y cualquier punto de llegada j más abajo en el río. Puede suceder que un viaje de i a j sea más caro que una sucesión de viajes más cortos, en cuyo caso se tomaría un primer bote hasta k y otro bote para continuar el viaje a partir de k (no hay coste adicional por cambiar de bote). Escribir un algoritmo eficiente para determinar el coste mínimo (así como las correspondientes paradas) para ir desde i hasta j y calcular el tiempo en función de N.

Ejercicio 7 Un archipiélago consta de unas cuantas islas y varios puentes que unen ciertos pares de islas entre sí. Para cada puente (que puede ser de dirección única o doble), además de saber la isla de origen y la de destino, se conoce su anchura > 0. La anchura de un camino formado por una sucesión de puentes es la anchura mínima entre las anchuras de los puentes que lo forman. Para

cada isla, se desea saber cuál es el camino de anchura máxima que las une (siempre que exista alguno).

Ejercicio 8 El Tio Antonio tiene en su granja dos vacas: *Devoradora* y *Listilla*. Antes de ordeñarlas las alimenta llenando un hilera de n cubos con pienso, siendo n par. Cada cubo i tiene una cantidad de pienso p_i distinta e indicada en el cubo. En su turno, cada vaca ha de elegir un cubo de uno de los extremos y comerse su contenido. El cubo se retira y el turno pasa a la otra vaca. Se sigue así hasta agotar los cubos. La vaca que comienza comiendo se determina por un procedimiento cualquiera. El objetivo de ambas vacas es comer en total lo máximo posible. La estrategia de *Devoradora* consiste en escoger el cubo de un extremo que esté más lleno. En cambio *Listilla* utiliza un algoritmo de programación dinámica de coste lineal. Diseñar tal algoritmo que ha de garantizar que, siempre que comience comiendo ella, come al menos tanto como *Devoradora*, independientemente de la estrategia que siga esta última.

Ejercicio 9 Jaimito, Juanito y Jorgito quieren construir una cabaña de exploradores en la playa. Con el objetivo de protegerla de la marea, van a levantar cuatro pilares sobre los que construirla y para ello disponen de un montón de ladrillos. Ahora bien, los ladrillos proceden de muy diversas fábricas y sus alturas no son las mismas. El objetivo de los jóvenes castores es determinar la distribución de los ladrillos para que el pilar más pequeño sea tan alto como sea posible. Suponiendo que las alturas de los ladrillos son valores enteros, resolver el problema mediante programación dinámica, indicando costes en tiempo y espacio.

Ejercicio 10 El grado de relación de dos genes se mide en función de hasta qué punto se pueden alinear. Para formalizar esta idea, piensa en un gen como en una cadena sobre el alfabeto $\Sigma = \{A, C, G, T\}$. Considera dos genes, $x = ATGCC$ e $y = TACGCA$. Una alineación de x e y es una forma de emparejar estas dos cadenas escribiéndolas en columnas; por ejemplo:

$$\begin{array}{c} -AT-GCC \\ TA-CGCA \end{array}$$

El guión "$-$" señala un hueco. Los caracteres de cada cadena deben aparecer en orden y cada columna debe contener un carácter de al menos una de las dos cadenas. La puntuación de un alineamiento se calcula sumando las puntuaciones de los pares emparejados, utilizando para ello una matriz de puntuación P de tamaño 5×5, donde la columna y la fila extras se utilizan para acomodar los huecos. Por ejemplo, el alineamiento anterior tiene la siguiente puntuación:

$$P[-,T] + P[A,A] + P[T,-] + P[-,C] + P[G,G] + P[C,C] + P[C,A]$$

Escribir un algoritmo de programación dinámica que, dadas dos cadenas $x[1..n]$ y $y[1..m]$ y una matriz de puntuación P devuelva el alineamiento de mayor puntuación.

Ejercicio 11 Un procesador de textos similar a LaTeX ha de decidir por dónde ha de partir las páginas en un texto formado por n párrafos consecutivos cuyas alturas (enteras) en milímetros son $l_1, \ldots, l_n$. Dichas alturas incluyen ya los interlineados necesarios para separar los párrafos. Conocemos la altura máxima P que puede ocupar el texto en cada página. No se puede alterar el orden de los párrafos, ni se puede partir un párrafo entre dos páginas. El objetivo es dedicar cada página a una secuencia de párrafos de manera que los sobrantes en cada página intermedia sean al tiempo pequeños y lo más homogéneos que sea posible. En concreto, si en una página se deja un espacio en blanco final de altura l, incurriremos en una penalización l^3. Esta penalización no se aplica a la última página. Diseñar e implementar un algoritmo que decida la forma óptima de repartir los párrafos en páginas, entendiendo por tal aquella en la que la suma de las penalizaciones sea mínima.

Ejercicio 12 Una de las pruebas habituales del concurso *Supervivientes* consiste en procurarse de manera autónoma la alimentación. A los concursantes se les facilitan m pequeñas huertas $h_1, \ldots, h_m$ y tendrán qué decidir cómo invertirán en ellas los n días de trabajo disponibles. Saben el beneficio $b(i, d)$ que sacarán de cada huerta h_i en función del número de días d que trabajen en ella. Obsérvese que es posible obtener un beneficio $b(i, 0) > 0$ aunque no se trabaje ningún día en esa huerta. Encontrar la asignación de días a huertas que maximiza el beneficio total.

Ejercicio 13 El carpintero Ebanisto recibe el encargo de cortar un tronco en n trozos, con cortes que han sido previamente marcados sobre la madera. Llamaremos $l_1, \ldots, l_{n+1}$ a las longitudes de los fragmentos resultantes tras hacer los n cortes establecidos. Sabemos que el esfuerzo de cortar cada fragmento de tronco en dos es proporcional a su longitud. Ebanisto se da cuenta de que el orden en que realice los cortes influye en el esfuerzo total empleado en el proceso de corte del tronco.

Diseñar e implementar un algoritmo que indique a Ebanisto un orden posible en el que debe realizar los cortes para que el esfuerzo total sea mínimo. Indicar en particular los tipos de datos concretos que se utilizan en la implementación. Analizar el coste en tiempo y en espacio del algoritmo propuesto.

Ejercicio 14 Se dispone de dos vasijas con capacidades enteras distintas C_1 y C_2, dadas en litros, y de una fuente de agua ilimitada. Se desea alcanzar en una de ellas una cantidad exacta de litros de agua C, para lo cual podemos realizar acciones de llenado y vaciado de una vasija, o de trasvase de agua entre las dos vasijas, sin derramar nada. En todo trasvase de una vasija A a otra B, o bien B se llena completamente, o bien A se vacía completamente (o bien ambas cosas a la vez). Por ejemplo, si $C_1 = 7$ litros y $C_2 = 11$ litros, es posible conseguir en no más de 10 acciones la cantidad de 6 litros en una de ellas. Se desea saber si el problema tiene solución en a lo sumo n acciones elementales.

Diseñar un algoritmo que, dados C, C_1, C_2 y n arbitrarios, diga si el problema tiene solución, y en caso afirmativo proporcione una secuencia de acciones elementales para alcanzarla. Solo deben usarse las ideas de programación dinámica, sin recurrir a ninguna pericia que maneje la teoría de números.

Ejercicio 15 Dado un grafo dirigido $G = (V, A)$ con costes en las aristas, resolver mediate programación dinámica el problema del viajante de comercio. Se ha de dar el detalle de todos los pasos: recurrencia, solución iterativa, recuperación del ciclo hamiltoniano de mínimo coste, y coste del algoritmo en tiempo y en espacio.

Ejercicio 16 Nos dan una secuencia $x_1, \ldots, x_n$ de n números naturales cuyo orden no podremos alterar. Se pide decidir si es posible intercalar entre cada dos de ellos los signos $+$ o $-$, de manera que el valor de la expresión resultante, evaluada de izquierda a derecha, y sin ningún tipo de precedencia entre sus operadores, sea finalmente 0. Por ejemplo, dada la secuencia $8, 2, 4, 3, 1$ podríamos en este caso obtener 0 así: $8 - 2 - 4 - 3 + 1 = 6 - 4 - 3 + 1 = 2 - 3 + 1 = -1 + 1 = 0$.

Indicación: Como paso previo a la resolución, y fijada la secuencia dada, se sugiere calcular el valor máximo que podría obtenerse al evaluar cualquiera de las expresiones resultantes.

Ejercicio 17 Sean $X = x_1, \ldots, x_n$ e $Y = y_1, \ldots, y_m$ dos secuencias formadas ambas con letras del alfabeto $\{A, C, G, T\}$, que representan dos hebras de ADN que se desea comparar. En concreto, se pretende encontrar alguna subsecuencia común a ambas secuencias de la máxima longitud posible. Entendemos por subsecuencia de una secuencia dada X a toda secuencia que resulte de eliminar de X cero o más de sus letras. Por ejemplo, $\langle C, G, A, C\rangle$ es una subsecuencia de $\langle T, A, C, A, G, C, A, C\rangle$.

Desarrollar un algoritmo de programación dinámica que resuelva el problema. Indicar la complejidad del mismo, tanto en tiempo como en memoria.

Ejercicio 18 Durante la fiesta de Halloween, y con el fin de recaudar dinero para el viaje de fin de curso, Alicia ha de vender P prendas a sus vecinos para que se confeccionen sus disfraces. Puede vender a cada uno de ellos un número variable k de prendas, con $0 \leq k \leq h$, siendo h una constante dada, y para cada vecino, $i \in \{1 \cdots n\}$, Alicia sabe qué beneficio b_{ik} obtendría si le vendiera k prendas.

Implementar un algoritmo que nos diga cuántas prendas habrá de vender Alicia a cada uno de sus vecinos, de forma que el beneficio total obtenido sea máximo. Indicar justificadamente su coste en tiempo y en espacio.

Ejercicio 19 Diseñar un algoritmo que encuentre el máximo valor que puede obtenerse colocando paréntesis de forma apropiada en la expresión

$$x_1/x_2/x_3/\cdots/x_{n-1}/x_n,$$

donde $x_1, x_2, \ldots, x_n$ son números reales positivos y '/' denota su división. El algoritmo debe mostrar también la forma en la que los paréntesis deben colocarse para obtener dicho valor máximo. Por ejemplo, si la expresión es $3/1/2$, colocando los paréntesis de la forma $(3/1)/2$ obtenemos $1,5$ mientras que colocando los paréntesis de la forma $3/(1/2)$ obtenemos 6.

Ejercicio 20 Nos dan un polígono P convexo de n vértices, numerados $1, \ldots, n$ en sentido horario. Queremos descomponerlo en triángulos, trazando suficientes diagonales de forma que estas no se corten entre sí. Nos dan también una matriz d simétrica $n \times n$ tal que d_{ij} contiene la longitud de la diagonal que une los vértices i y j. Por conveniencia, si esos dos vértices son el mismo, o son adyacentes—es decir, no forman una diagonal, sino un lado del polígono—, la matriz contiene $d_{ij} = 0$. Desarrollar un algoritmo que calcule la descomposición en triángulos de un polígono P dado, cuya suma de longitudes de sus diagonales sea mínima.

Ejercicio 21 La empresa *SlowTravel* ofrece un recorrido en burro por la Alcarria. La ruta consiste en visitar n pueblos en un orden preestablecido $1, 2, \ldots, n$. En cada pueblo, el viajero tiene la posiblidad de cambiar de burro o de seguir con el que tiene. Se conoce el coste p_{ij} de alquilar un pollino en cada pueblo i y dejarlo en cualquier pueblo j situado más adelante en la ruta. Desarrollar un algoritmo que encuentre la secuencia de alquileres de pollinos para hacer la ruta completa, que tenga coste mínimo. Los costes en tiempo y en espacio del algoritmo no han de exceder respectivamente $O(n^2)$ y $O(n)$.

Ejercicio 22 Dada una secuencia no vacía de enteros $x_1, \ldots, x_n$, denominamos **subsecuencia** a cualquier secuencia no vacía de dichos enteros, siempre que aparezcan en posiciones consecutivas de la secuencia original. Diseñar y programar por programación dinámica los siguientes dos problemas:

1. Una función que, dada una posición i, devuelva la subsecuencia de suma máxima acabada en x_i.
2. Una función que devuelva la subsecuencia de suma máxima de la secuencia original. Puede utilizarse para ello la función desarrollada en el apartado anterior.

Ejercicio 23 Dado un conjunto $\{v_1, \ldots, v_n\}$ de números naturales, diseñar un algoritmo que decida cómo partirlo en dos subconjuntos de idéntica suma, o que indique que tal partición no es posible. Dar el coste en tiempo y en espacio del algoritmo.