◁ Ejercicio 1 ▷

Sea V un vector con n valores enteros distintos. Diseñe un algoritmo eficiente basado en programación dinámica para encontrar la secuencia creciente de máxima longitud en V. Por ejemplo, si el vector de entrada es $(11, 17, 5, 8, 6, 4, 7, 12, 3)$, la secuencia creciente de máxima longitud es $(5, 6, 7, 12)$.

◁ Ejercicio 2 ▷

Diseñe un algoritmo basado en programación dinámica para resolver el problema del viajante de comercio en tiempo $O(n^2 \cdot 2^n)$. ¿Es este algoritmo mejor que un algoritmo que explore todas las permutaciones posibles? Como sugerencia, defina $g(i, S)$ como la longitud del camino más corto que des el vértice i hasta el nodo 1 (donde comienza el circuito) que pase exactamente una vez por cada vértice del conjunto S. Con esta definición, dado un grafo $G = (V, A)$, $g(1, V - \{1\})$ nos da la longitud del circuito hamiltoniano minimal.

◁ Ejercicio 3 ▷

Sea el alfabeto $\Sigma = \{a, b, c\}$. Los elementos de Σ tienen la siguiente tabla de multiplicación, donde las filas muestran el primer operando y las columnas muestran el segundo operando de la multiplicación de símbolos:

	a	**b**	**c**
a	b	b	a
b	c	b	a
c	a	c	c

Esto es, $a \cdot b = b$, $b \cdot a = c$, y así sucesivamente. Diseñe un algoritmo eficiente basado en programación dinámica que examine una cadena de caracteres $x = x_1 x_2 \ldots x_n$ y decida si es posible o no poner paréntesis en x de tal manera que el valor de la expresión resultante sea a.

◁ Ejercicio 4 ▷

A lo largo de un río hay n aldeas. En cada aldea, se puede alquilar una canoa que se puede devolver en cualquier otra aldea que esté a favor de corriente (resulta casi imposible remar a contra corriente). Para todo posible punto de partida i y para todo posible punto de llegada j, se conoce el coste de alquilar una canoa desde i hasta j. Sin embargo, puede ocurrir que el coste de alquiler desde i hasta j sea mayor que el coste total de una serie de alquileres más breves. En tal caso, se puede devolver la primera canoa en alguna aldea k situada entre i y j y seguir el camino en una nueva canoa (no hay costes adicionales por cambiar de canoa de esta manera). Diseñe un algoritmo basado en programación dinámica para determinar el coste mínimo del viaje en canoa desde todos los puntos posibles de partida i a todos los posibles puntos de llegada j. Aplique el algoritmo diseñado en la resolución del siguiente caso:

	1	**2**	**3**	**4**	**5**
1	0	17	8	16	20
2	∞	0	12	6	15
3	∞	∞	0	12	16
4	∞	∞	∞	0	15
5	∞	∞	∞	∞	0

◁ Ejercicio 5 ▷

Se tienen n unidades de un recurso que deben asignarse a r proyectos. Si se asignan j, $0 \leq j \leq n$, unidades al proyecto i, se obtiene un beneficio $N_{i,j}$. Diseñe un algoritmo que asigne recursos a los r proyectos minimizando el beneficio total obtenido.

◁ Ejercicio 6 ▷

Dadas n funciones $f_1, f_2, \ldots, f_n$ y un entero positivo M, se desea maximizar la función $f_1(x_1) + f_2(x_2) + \ldots f_n(x_n)$ sujeta a la restricción $x_1 + x_2 + \ldots + x_n = M$, donde $f_i(0) = 0$ $(i = 1, \ldots, n)$, x_i son números naturales, y todas las funciones son monótonas crecientes, es decir, $x \geq y$ implica que $f_i(x) > f_i(y)$. Suponga que los valores de cada función se almacenan en un vector.

◁ Ejercicio 7 ▷

La función de Ackermann se define recursivamente del modo siguiente:

$$\begin{cases} Ack(0,n) = n+1 & \\ Ack(m,0) = Ack(m-1,1) & \text{si } m > 0 \\ Ack(m,n) = Ack(m-1, Ack(m,n-1)) & \text{si } m,n > 0 \end{cases}$$

Diseñe un algoritmo de programación dinámica para calcular $Ack(m,n)$.

◁ Ejercicio 8 ▷

Una empresa de ferrocarriles sirve n estaciones $S_1, \ldots, S_n$ y trata de mejorar su servicio al cliente mediante terminales de información. Dadas una estación origen S_0 y una estación destino S_d, un terminal debe ofrecer (inmediatamente) la información sobre el horario de los trenes que hacen la conexión entre S_0 y S_d y que minimizan el tiempo de trayecto total. Se necesita diseñar un algoritmo que realice esta tarea a partir de la tabla con los horarios, suponiendo que las horas de salida de los trenes coinciden con las de sus llegadas (es decir, que no hay tiempos de espera) y que, naturalmente, no todas las estaciones están conectadas entre sí por líneas directas. Así, en muchos casos, hay que hacer transbordos aunque se supone que tardan tiempo cero en efectuarse.

◁ Ejercicio 9 ▷

Dados n objetos, se quiere calcular el número de ordenaciones posibles según las relaciones $<$ e $=$. Por ejemplo, dados tres objetos A, B y C, el algoritmo debe determinar que existen 13 ordenaciones distintas: $A = B = C$, $A = B < C$, $A < B = C$, $A < B < C$, $A < C < B$, $A = C < B$, $B < A = C$, $B < A < C$, $B < C < A$, $B = C < A$, $C < A = B$, $C < A < B$, y $C < B < A$.

◁ Ejercicio 10 ▷

Sean u y v dos cadenas de caracteres. Se desea transformar u en v con el mínimo número de operaciones básicas del tipo siguiente: eliminar un carácter, añadir un carácter, y cambiar un carácter. Por ejemplo, podemos pasar de $abbac$ a $abcbc$ en tres pasos: eliminar b en la posición 3 ($abac$), añadir b en la posición 4 ($ababc$), cambiar a en la posición 3 por c ($abcbc$). Sin embargo, esta transformación no es óptima. Lo que se quiere, en este caso, es diseñar un algoritmo que calcule el número mínimo de operaciones, de esos tres tipos, necesarias para transformar u en v y cuáles son esas operaciones, estudiando su complejidad en función de las longitudes de u y v.