**PROGRAMACIÓN DINÁMICA**
RELACIÓN DE EJERCICIOS Y PROBLEMAS

1. Diseñe algoritmos que permitan resolver eficientemente el problema de la mochila 0/1 para los siguientes casos:

   a. Mochila de capacidad W=15:

| Objeto | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Peso | 3 | 7 | 4 | 2 | 1 | 6 |
| Beneficio | 12 | 3 | 7 | 4 | 3 | 8 |

   b. Mochila de capacidad W=255:

| Objeto | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Peso | 51 | 119 | 68 | 34 | 17 | 102 |
| Beneficio | 204 | 51 | 119 | 68 | 51 | 136 |

   c. Mochila de capacidad W=1000:

| Objeto | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Peso | 130 | 570 | 140 | 200 | 360 | 400 |
| Beneficio | 120 | 300 | 570 | 423 | 300 | 800 |

   d. ¿Se podría utilizar programación dinámica cuando el peso de un objeto viene dado por un valor real en vez de entero?

   e. Un caso concreto del problema de la mochila puede tener más de una solución óptima: ¿cómo podríamos darnos cuenta de esto? Diseñe un algoritmo que devuelva todas las soluciones óptimas para un caso concreto del problema de la mochila 0/1.

2. Tenemos un alfabeto $\Sigma = \{a,b,c.d\}$ y una relación de orden total $\Phi$ definida sobre los elementos del alfabeto $\Sigma$. Sea $f(x,y)$ una función que recibe como entradas dos elementos de $\Sigma$ y nos indica si $x$ es el elemento que precede inmediatamente a $y$ en $\Phi$. Diseñe un algoritmo basado en programación dinámica que, dados dos elementos cualesquiera del alfabeto, $u$ y $v$, nos permita determinar si $u$ precede a $v$ en el orden $\Phi$.

   NOTA: En la relación de orden se verifica la propiedad transitiva.

**3. Problema del play-off:**

Supongamos que dos equipos de baloncesto, Informáticos CB y Basket Telecom, disputan un play-off que ganará el primero en conseguir n victorias (esto es, tendrán que disputar, como mucho 2n-1 partidos):

a. Si suponemos que la probabilidad de que Informáticos CB gane un partido concreto es un valor fijo P (p.ej. ½), calcule la probabilidad de que Informáticos CB gane el playoff.

b. Si suponemos que la probabilidad de que un equipo gane un partido concreto depende de si juega como equipo local o como visitante, siendo PL la probabilidad de que gane el partido como local y PV la probabilidad de que lo gane como visitante, con PL>PV, diseñe un algoritmo que permita determinar la influencia que supone sobre el resultado final el hecho de contar con ventaja de campo (esto es, jugar el primer partido del play-off en casa).

c. ¿Hay alguna diferencia si se intercalan los partidos en casa y fuera con respecto a jugar el primero en casa, los dos siguientes fuera, otros dos en casa y así sucesivamente?

CONSEJO: Utilice P(i,j) para representar la probabilidad de que el primer equipo gane el play-off si necesita ganar i partidos cuando el segundo equipo aún tendría que ganar j partidos. Antes de comenzar el play-off, por tanto, la probabilidad de que el primer equipo gane el play-off es P(n,n).

**4. Secuencia creciente de máxima longitud:**

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).

**5. Problema del viajante de comercio**

Diseñe un algoritmo basado en programación dinámica para resolver el problema del viajante de comercio en tiempo $O(n^2 2^n)$. ¿Es este algoritmo mejor que un algoritmo que explore todas las permutaciones posibles?

SUGERENCIA: Defina g(i,S) como la longitud del camino más corto que desde el nodo i hasta el nodo 1 (donde comenzamos nuestro circuito) que pase exactamente una vez por cada nodo del conjunto S. Con esta definición, dado un grafo G(V,E), g(1,V-{1}) nos da la longitud del circuito hamiltoniano minimal.

**6. "Multiplicación" de símbolos:**

Tenemos un alfabeto $\Sigma = \{a,b,c\}$. Los elementos de $\Sigma$ tienen la siguiente tabla de multiplicación, donde las filas muestran 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, ab = b, ba = c, y así sucesivamente. Diseñe un algoritmo eficiente basado en programación dinámica que examine una cadena de caracteres $x = x_1x_2 \ldots x_n$ y decida si es posible o no poner paréntesis en x de tal manera que el valor de la expresión resultante sea a.

**7. Problema de la cadena:**

Tenemos un conjunto de n eslabones ($e_1, e_2, \ldots, e_n$) que nos permiten construir una cadena. Para formar la cadena, tenemos que unir cada eslabón $e_i$ con el $e_{i+1}$, con i = 1.. n-1. Asociado a cada eslabón, tenemos el peso del mismo ($p_1, p_2, \ldots, p_n$). La cadena la vamos formando agrupando iterativamente los distintos eslabones en subcadenas.

Supongamos que el coste de agrupar dos subcadenas $c_i$ y $c_j$, $COSTE(c_i,c_j)$, es igual a la suma de los pesos de los eslabones en los extremos de las mismas.

Por ejemplo, dadas las cadenas

$c_1 = (e_i - e_{i+1} - \ldots - e_{k-1} - e_k)$,
$c_2 = (e_{k+1} - e_{k+2} - \ldots - e_m)$,
$c_3 = (e_{m+1})$,

podemos calcular

$COSTE(c_1, c_2) = p_i + p_k + p_{k+1} + p_m$
$COSTE(c_2,c_3) = p_{k+1} + p_m + p_{m+1}$

Para unir los n eslabones de una cadena, por tanto, serán necesarias n-1 uniones.

Definiendo el coste total para formar la cadena como la suma de los costes necesarios en cada una de estas uniones, diseñe un algoritmo que nos permita obtener la forma de construir la cadena que tenga menor coste total.

**8. Alquiler de canoas:**

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).

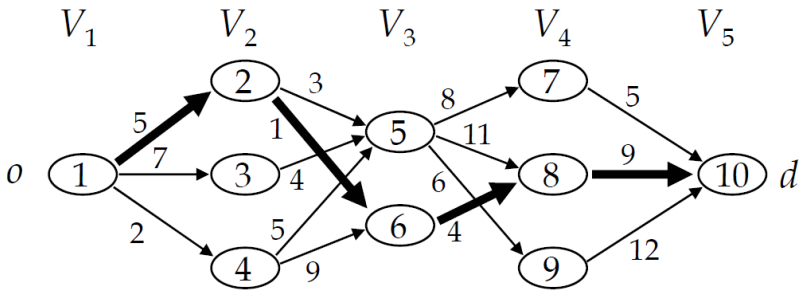
Para todo posible punto de partida i y para todo posible punto de llegada j, se conoce el coste de alquilar una canoa desde i hasta j. Sin embargo, puede ocurrir que el coste del alquiler desde i hasta j sea mayor que el coste total de una serie de alquileres más breves. En tal caso, se puede devolver 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 dicho algoritmo 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 |

**9. Caminos mínimos en grafos multietapa:**

Un grafo multietapa G=(V,E) es un grafo dirigido en el que se puede hacer una partición del conjunto V de vértices en k ($k \geq 2$) conjuntos distintos $V_i$, $1 \leq i \leq k$, tal que, para todo arco del grafo (u,v), $u \in V_i$ y $v \in V_{i+1}$ para algún i, $1 \leq i < k$. Los conjuntos $V_1$ y $V_k$ tienen un único vértice: los vértices origen y destino del grafo.




Teniendo en cuenta que todo camino entre los vértices origen y destino tiene exactamente un vértice en cada etapa $V_i$, diseñe un algoritmo que encuentre el camino de coste mínimo entre los vértices origen y destino.

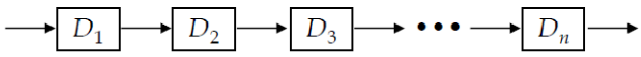
**10. Asignación de recursos:**

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 $N_{i,j}$.

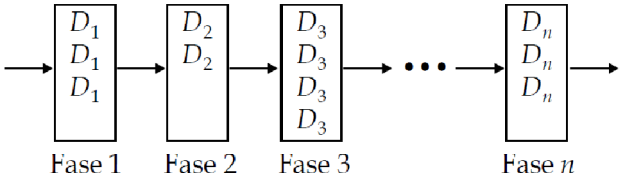
Diseñe un algoritmo que asigne recursos a los r proyectos maximizando el beneficio total obtenido.

**11. Diseño de sistemas fiables:**

Se desea diseñar un sistema compuesto de varios dispositivos conectados en serie:




Sea $r_i$ la fiabilidad del dispositivo $D_i$, medida como la probabilidad de que funcione correctamente en un instante de tiempo dado. Entonces, la fiabilidad del sistema completo es el producto de la fiabilidad de los dispositivos conectados en serie. Por ejemplo, si n=10 y $r_i$=0.99, $1 \leq i \leq 10$, la fiabilidad del sistema es $\Pi r_i = 0.904$.




Una forma de aumentar la fiabilidad es colocar varios dispositivos en paralela en cada etapa de dispositivos en serie. Si una fase i contiene $m_i$ copias del dispositivo $D_i$, la probabilidad de que toda la fase falle es $(1-r_i)^m$, por lo que su fiabilidad será $1-(1-r_i)^m$. Por ejemplo, si $r_i$=0.99 y $m_i$=2, la fiabilidad de la fase i pasará de 0.99 a 0.9999.

En realidad, la fiabilidad de la fase i es algo menor que $1-(1-r_i)^m$, ya que las copias de un mismo dispositivo no son completamente independientes.

Si denotamos la fiabilidad de la fase i por $\Phi(m_i)$, la fiabilidad del sistema será $\Pi_{1 \leq i \leq n} \Phi(m_i)$. Diseñe un algoritmo que maximice la fiabilidad del sistema suponiendo que nos imponen una limitación sobre su coste C, esto es

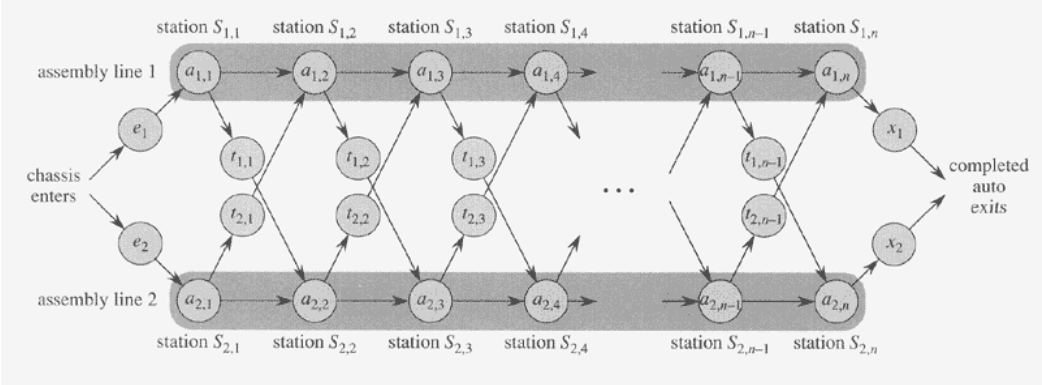
maximizar $\Pi_{1 \leq i \leq n} \Phi(m_i)$

sujeto a $\Sigma_{1 \leq i \leq m} c_i m_i \leq C$

donde $c_i$ es el coste de cada unidad del dispositivo de la fase i.

## 12. Cadenas de montaje:

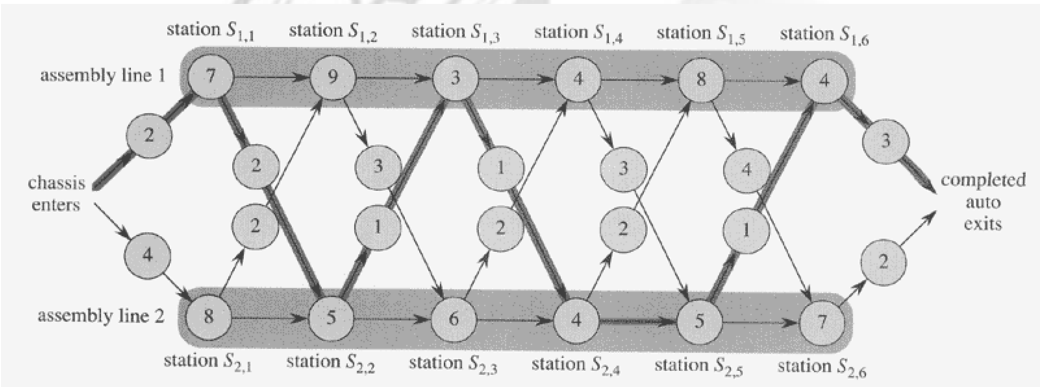
En una fábrica de coches disponemos de dos cadenas de montaje, cada una de ellas con n estaciones:




La estación j de la cadena i se denomina $S_{i,j}$ y necesita $a_{i,j}$ minutos para realizar su trabajo. Un chasis comienza su recorrido por la fábrica en una de las cadenas de montaje. Tras su paso por la estación j, un chasis puede pasar a la estación j+1 de cualquiera de las cadenas de montaje. El cambio de una estación a otra no conlleva ningún coste adicional si el chasis se mantiene en la misma cadena, pero consume un tiempo $t_{i,j}$ en pasar de la estación $S_{i,j}$ a la otra cadena de montaje. Al llegar al final de la cadena de montaje i, el coche tarda un tiempo $x_i$ en salir de la fábrica.

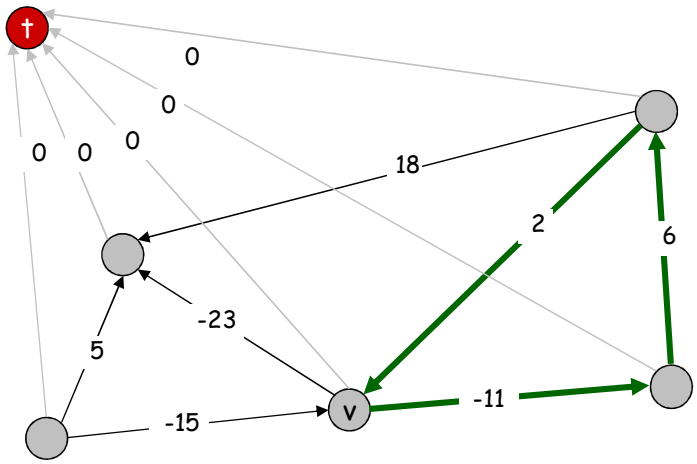
El problema consiste en determinar qué estaciones utilizar de cada cadena de montaje para minimizar el tiempo total necesario para producir un vehículo cuando nuestra fábrica recibe un pedido urgente. Diseñe un algoritmo que resuelva este problema.

Resuelva la siguiente instancia del problema:



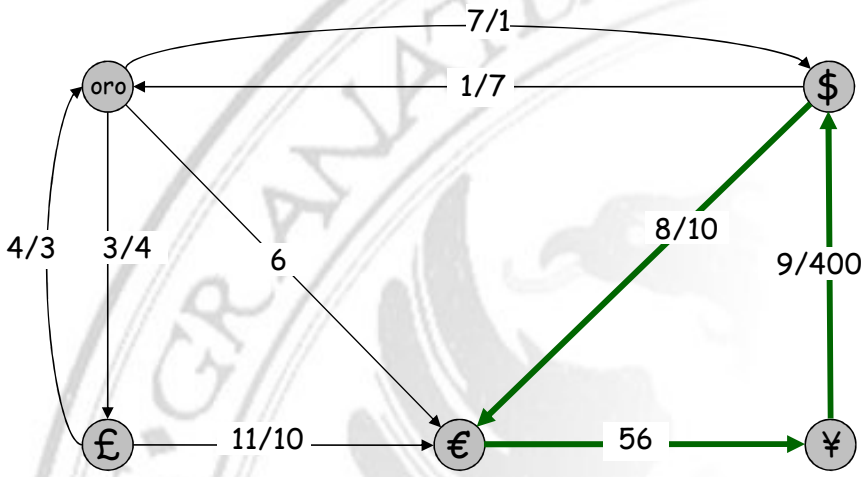
## 13. Ciclos negativos

Diseñe un algoritmo que nos permita detectar ciclos negativos en un grafo dirigido (sugerencia: utilizar los resultados obtenidos por el algoritmo de Bellman-Ford sobre el grafo resultante de añadir un nuevo nodo a nuestro grafo y arcos de coste 0 que unan cada uno de los nodos del grafo con el nuevo nodo).




## 14. Arbitraje en el mercado de divisas:

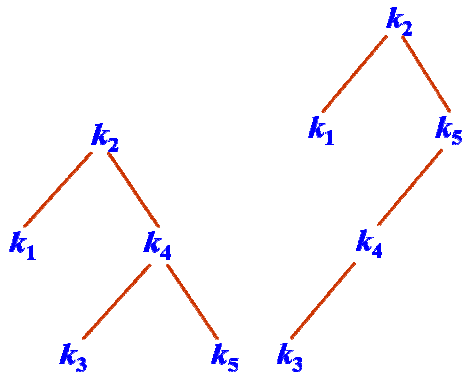
Dados un conjunto de monedas y sus tipos de cambio actuales, identificar si existe alguna posibilidad de arbitraje que nos reporte un beneficio:




Esto es, podemos comprar 5600 yenes con 100 euros, que luego podemos cambiar por 5600*9/400 = 126 dólares, que luego podemos convertir de nuevo en 126 * 8/10 = 100.80 euros, obteniendo un beneficio del 0.8% en la operación.

## 15. ABB óptimos (con probabilidades de acceso no uniforme):

Dados n elementos $k_1 < k_2 < ... < k_n$, con probabilidades de acceso conocidas $p_1$, $p_2$, ..., $p_n$, se desea encontrar el árbol binario de búsqueda que minimice el coste esperado de búsqueda (en términos del número de comparaciones necesarias para devolver un resultado).




| | $p_i$ | $c_i$ | $c_i \cdot p_i$ | $c_i$ | $c_i \cdot p_i$ |
|---|---|---|---|---|---|
| $k_1$ | 0.25 | 2 | 0.50 | 2 | 0.50 |
| $k_2$ | 0.20 | 1 | 0.20 | 1 | 0.20 |
| $k_3$ | 0.05 | 3 | 0.15 | 4 | 0.20 |
| $k_4$ | 0.20 | 2 | 0.40 | 3 | 0.60 |
| $k_5$ | 0.30 | 3 | 0.90 | 2 | 0.60 |
| **coste** | | | **2.15** | | **2.10** |

NOTA: El árbol de la derecha es el óptimo para este conjunto de claves.

VARIANTE
Dados n elementos $k_1 < k_2 < ... < k_n$, con probabilidades de acceso conocidas $p_1$, $p_2$, ..., $p_n$, y con probabilidades de búsqueda infructuosa conocidas $q_0$, $q_1$, ..., $q_n$, se desea encontrar el ABB que minimice el coste esperado de búsqueda.