

- 1.– Programe el Problema del Viajante utilizando funciones con memoria. Se ha de obtener un procedimiento incluido en una unidad al que proporcionándole la matriz de costes del grafo, nos devuelva en la solución la vuelta de mínimo coste y su coste asociado.
- 2.– 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
 - cambiar un carácter

Por ejemplo, para pasar de la cadena $ababc$ a $bccba$ podríamos hacer:

$ababc \rightarrow babc$ (eliminar a)
 $\rightarrow bcabc$ (añadir c)
 $\rightarrow bccbc$ (cambiar un carácter)
 $\rightarrow bccba$ (cambiar un carácter)

Escriba un algoritmo de Programación Dinámica que calcule el número mínimo de operaciones necesarias para transformar u en v y cuáles son esas operaciones. Implementelo.

- 3.– Sobre el río Tajo hay N estafetas de correos. En cada estafeta se puede alquilar un bote que permite ir a cualquier otra estafeta río abajo (es imposible remontar la corriente). La tarifa indica el coste del viaje de i a j para cualquier punto de partida i y cualquier punto de llegada j más abajo del 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 una estafeta k y un segundo bote para continuar a partir de k . No hay coste adicional por cambiar de bote. Diseñe e implemente un algoritmo eficiente que determine el coste mínimo para ir de i a j .
- 4.– **Problema Simplificado de Transporte.**

Se tienen dos almacenes A_1 y A_2 , y n comercios c_1, c_2, \dots, c_n . En cada almacén hay e_1 y e_2 , respectivamente, unidades indivisibles de un producto; y en cada comercio se necesitan v_j ($j = 1, 2, \dots, n$). Además, se verifica:

$$e_1 + e_2 = \sum_{i=1}^n v_i$$

El coste de transporte de A_i a C_j viene dado por la función $Coste_{ij}(x)$, donde x es el número de unidades transportadas.

Se pide la solución óptima, i.e., de coste mínimo, para el problema de abastecimiento a los comercios, por un método de **Programación Dinámica**.

- 5.– Resuelva el Problema de la Mochila versión 0/1 utilizando **Backtracking**.
- 6.– N trabajos han de procesarse en un sistema que cuenta con dos procesadores. Para cada trabajo se saben los tiempos a_i y b_i que necesitaría el proceso en cada procesador. Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que encuentre la solución óptima, esto es, la que minimiza el tiempo necesario para concluir todos los procesos.
- 7.– El **Centro de un grafo** se define como el vértice del grafo que minimiza la distancia media de todos los vértices del grafo a dicho vértice. Diseñe un algoritmo que halle el centro de un grafo. Calcule la complejidad del algoritmo desarrollado.
- 8.– Construya e implemente un algoritmo que dada una sucesión finita encuentre una de las subsucesiones crecientes más largas (puede haber varias), utilizando bien **Programación Dinámica** bien **Backtracking**.
- 9.– La partición de un número positivo N consiste en escribir ese número como la suma de k números positivos, por supuesto distintos del original. Por ejemplo:

$$12 = 1 + 1 + 2 + 3 + 5 = 1 + 2 + 4 + 5 = 1 + 2 + 2 + 2 + 5$$

Construya un programa que genere todas las posibles particiones de un número natural dado.

- 10.– Unos niños juegan en una calle adoquinada a la rayuela. Situados en un adoquín pueden desplazarse al siguiente o saltárselo. Construya un procedimiento que imprima todos los caminos posibles para situados en el primer adoquín ir al n -ésimo. ¿Cuántos caminos hay para alcanzar dicho adoquín?
- 11.– Problema de **Asignación de Trabajos**.

N trabajos han de ser realizados por N personas, y se sabe que la persona i tardará un tiempo C_{ij} , en hacer el trabajo j .

Resuelva los siguientes problemas utilizando **Backtracking**:

- Cada trabajo debe hacerlo una persona distinta y se debe minimizar el tiempo total.
- Como en el apartado a) pero sin la restricción de que cada persona deba hacer un trabajo.
- Como en a) pero minimizando el máximo tiempo para concluir los trabajos (todos los trabajadores comienzan simultáneamente).

- 12.– En un Centro de Salud, un día normal, tenemos P pacientes y M médicos ($M \leq P$). Un paciente sólo será visto por un médico. Tenemos una estimación del tiempo t_{mp} que tarda en atender el médico m al paciente p .

Supuesto que todos los médicos comienzan su consulta a la misma hora, diseñe algoritmos, cuidando la eficiencia, para resolver de los siguientes problemas:

- i) **Minimizar el tiempo total de atención a los pacientes.** (No hay ninguna restricción.)
- ii) Como en i) pero **cada médico ha de ver por lo menos a un paciente.**
- iii) Suponga ahora que los M médicos forman un equipo que se ha desplazado *ex profeso*, utilizando un transporte colectivo, a la localidad del Centro de Salud, y **desean regresar a su centro origen cuanto antes.** **NOTA:** Téngase presente que **no pueden regresar hasta que no se atienda a todos los pacientes**, o lo que es lo mismo, hasta que terminen de atender sus consultas todos los médicos.
- iv) Como en el apartado anterior, pero suponiendo además, que entre los M , hay K dados, que sistemáticamente se niegan a ver más de L pacientes cada uno, teniendo que ser asumidas las otras consultas por el resto.

- 13.– El ministerio de Desinformación y Decencia se ha propuesto hacer trabajar en firme a sus N funcionarios, y se ha sacado de la manga N trabajos. Los funcionarios, ya se sabe que son capaces de realizar cualquier trabajo, pero unos tardan todavía más que otros, y los hay que lo hacen aún pero que otros, lo que se recoge en dos tablas T y E , siendo $T_{f,t}$ el tiempo que tarda el funcionario f en realizar el trabajo t , y $E_{f,t}$ es la eficacia con la que el funcionario f realiza el trabajo t . Su Excelencia el Sr. Ministro, siempre tan preocupado por la eficiencia, desea encontrar la asignación óptima de trabajos a funcionarios en dos sentidos diferentes (e independientes):

- a) **minimizar la suma total de tiempos,**
- b) **maximizar la suma total de eficacias,**

Diseñe algoritmos **Backtracking** que resuelvan dichos problemas.

- 14.– Un cuadrado latino de orden n es un tablero de $n \times n$ posiciones, cada una de las cuales posee un color escogido entre n colores diferentes, tal que ni en ninguna de sus filas ni en ninguna de sus columnas hay colores repetidos.

Ejemplo ($n = 4$):

Rojo	Blanco	Negro	Azul
Blanco	Negro	Azul	Rojo
Azul	Rojo	Blanco	Negro
Negro	Azul	Rojo	Blanco

Diseñe e implemente un algoritmo que imprima todos los posibles tableros latinos de tamaño $n \times n$.

- 15.– Disponemos de una baraja de cartas española con cuatro palos (oros, copas, espadas y bastos). De cada uno de los palos tenemos las siguientes cartas: 1, 2, 3, 4, 5, 6, 7, 10, 11 y 12.

Diseñe e implemente un algoritmo que calcule la manera de obtener "siete y media" de modo que se minimice el número de cartas empleadas.

NOTA: Todas las cartas puntúan por su valor excepto 10, 11 y 12 que sólo valen medio punto.

- 16.– Sea S un conjunto de enteros positivos, se desea averiguar si existe una partición $\{P, Q\}$ disjunta de S , i.e., $P \cap Q = \emptyset$ y $P \cup Q = S$, tal que

$$\sum_{s \in P} s = \sum_{s \in Q} s$$

- 17.– Tenemos un conjunto S de N enteros positivos, se trata de hacer una partición disjunta del mismo en dos subconjuntos P y Q de tal forma que se minimice la diferencia de la suma de los elementos de cada subconjunto, i.e., se trata de hallar $\{P, Q\}$ que verifican $P \cap Q = \emptyset$ y $P \cup Q = S$, y tales que minimicen la expresión:

$$\left| \sum_{s \in P} s - \sum_{s \in Q} s \right|$$

Diseñe un algoritmo **Backtracking** que resuelva el problema.

- 18.– Lolo Luke, Director General de CAMP-MUSIC, nos ha encargado la difícil tarea de organizar las grabaciones de su poderosa compañía, de tal manera que exista una compensación de tiempos en sus grabaciones en cinta. Para ello, nos pide que diseñemos un algoritmo que tomando como datos los tiempos de todas las canciones que vayan a ser grabadas sean distribuidas en las dos caras. (De un Ángel).

- 19.– El presupuesto para comprar libros de una determinada organización es menor que la cantidad necesaria para la adquisición de todos los libros que deseamos comprar. Diseñe un algoritmo que nos indique los libros a comprar teniendo en cuenta lo siguiente:

- i) Se desea gastar la mayor parte posible del presupuesto disponible.
- ii) Jamás se podrá pagar un importe superior a la cantidad presupuestada.
- iii) No hay ningún libro preferido sobre otro.
- iv) Se dispone de la lista completa de libros que deseamos con sus correspondientes precios.

- 20.– Se tiene un tablero rectangular de altura N y anchura K , en cuyas casillas hay letras. Se pide un programa que utilizando Backtracking, encuentre la sucesión de vocales más larga que puede formarse partiendo del extremo inferior izquierdo y permitiéndose el acceso desde cada casilla a las adyacentes vertical, horizontalmente o en diagonal, sin pasar dos veces por una misma casilla.

Por otra parte si $K = 2$, existe un método mezcla de "*Método Devorador*" y "*Divide y Vencerás*" que permite resolver el problema. Constrúyalo y prográmelo.

- 21.– El emperador se dispone a expandir su imperio. Dispone de una lista de países para atacar, cada uno de los cuales tiene una resistencia r_i . Esta resistencia puede ser vencida con un número igual de soldados. Cada país proporciona un beneficio b_i en cuanto a recursos que dispone: oro, granjas, esclavos... Pero con cada país conquistado genera un nivel de descontento d_i en el ejército y si sobrepasa cierto límite el ejército puede sublevarse. Como todos sabemos el emperador no tiene inteligencia para solucionar el problema así que ha pedido ayuda a su lacayo, que debe diseñar un algoritmo que **maximice el beneficio que se consigue sin sobrepasar el número de soldados y sin que se subleve el ejército**.
- 22.– Tenemos una colección de n objetos que debemos empaquetar en envases de capacidad C . El objeto i tiene un volumen v_i . Diseñe e implemente un algoritmo que calcule el empaquetamiento óptimo, esto es, que minimice el número de envases a utilizar. Evidentemente, los objetos no se pueden fraccionar. Para simplificar el problema considere que un objeto puede incluirse en un envase si el espacio libre que queda en el mismo es mayor o igual que el tamaño del objeto, ignorando la forma que el objeto puede tener.
- 23.– Construya un programa que utilice el procedimiento anterior para hacer una copia de seguridad de los ficheros del disco duro de un directorio y sus subdirectorios en el menor número de discos flexibles necesarios.
- 24.– Tenemos N productos que debemos empaquetar en bolsas. Cada producto i tiene un volumen v_i y un peso p_i . Todas las bolsas tienen la misma capacidad V y la misma resistencia en peso P . Diseñe un algoritmo que nos devuelva el **número mínimo de bolsas necesarias para empaquetar los N objetos, junto con la distribución de los mismos en las bolsas**.
- 25.– *Papa Noel* tiene que repartir los juguetes a los niños y lo hará con una flota de trineos cada uno de los cuales puede soportar una carga máxima P_{max} (la misma para todos). Tenemos N juguetes, cada uno de los cuales tiene un peso dado. Diseñe un algoritmo ***Backtracking*** que distribuya los N juguetes entre los trineos de tal forma que se **minimice el número de trineos utilizados**.

- 26.– Disponemos de N de prendas y una lavadora. Cada prenda i tiene un volumen v_i y un peso p_i . El tambor de la lavadora tiene un volumen V y es capaz de soportar un peso P . Diseñe un algoritmo que **minimice el número de lavados** necesarios para lavar las N prendas.
- 27.– Disponemos de N de prendas y una lavadora. Cada prenda i tiene un volumen v_i y nos aporta un beneficio b_i si la lavamos. El tambor de la lavadora tiene un volumen V . Diseñe algoritmos que resuelvan los problemas siguientes:
- Se trata de llenar la lavadora de tal forma que se maximice el beneficio de las prendas lavadas, respetando la restricción de volumen.
 - Se trata de minimizar el número de lavados necesarios para lavar todas las prendas, respetando la restricción de volumen.
- 28.– Tenemos dos bolsas capaces de soportar hasta un peso máximo P_{max} , y un conjunto de N objetos de cada uno de los cuales conocemos su peso p_i . Se trata de decidir qué objetos se ponen en cada bolsa para que el peso en ambas esté lo más equilibrado posible, sin sobrepasar el peso máximo P_{max} , sabiendo que el peso de los objetos de cada bolsa debe ser al menos P_{min} ($P_{min} \leq P_{max}$). Diseñe un algoritmo que resuelva el problema basado en i) **Programación Dinámica** y ii) **Backtracking**. (Nótese que no necesariamente se deben incluir todos los objetos en las dos bolsas.)
- 29.– En el año 50 antes de Jesucristo toda la Galia está ocupada por los romanos... ¿Toda? ¡No! Una aldea poblada por irreductibles galos resiste todavía al invasor.

Tras años y años de escaramuzas con *César*, *Abracurcix*, el jefe de la aldea gala, ha decidido invitar a éste a un banquete en la aldea para firmar por fin la paz. Como homenaje a tan ilustre invitado *Abracurcix* ha encargado a sus mejores hombres, *Astérix* y *Obélix*, que construyan enseguida un pórtico de mármol a la entrada de la aldea para recibir a *César*.

Dado que la economía de la aldea no es muy boyante, el jefe le ha encargado a su cuñado que trabaja en la cantera, *Chapuzovic*, que le abastezca de los bloques de mármol necesarios para construir las dos columnas que componen el pórtico. Este último, aún molesto por no haber sido invitado al último banquete celebrado en la aldea, le ha dado los bloques defectuosos que no puede sacar a la venta por su baja calidad.

De este modo, *Astérix* y *Obélix* cuentan con N bloques de mármol de grosores distintos g_i . Tendrán que construir las dos columnas iguales, o si esto no es posible, de forma que haya la mínima diferencia entre ellas, a fin de que no se note mucho. El jefe también les ha informado de la normativa gubernamental vigente a cerca de las columnas de homenaje, que dice que no pueden ser más altas de una determinada altura H_{max} , y que tienen que tener un mínimo de H_{min} .

Como *Panoramix* está de fin de semana, *Algoritmix*, el druida suplente de la aldea, les ha indicado que podrían aplicar una estrategia basada en **Programación Dinámica** para resolver su problema. Como no tienen claro a qué se refería el druida con eso de “Programación

Dinámica”, debe ayudar a nuestros intrépidos galos a salir victoriosos una vez más de tan importante misión.

- 30.– A los profesores de Ampliación de Programación los cambios de estación les producen extraños efectos, en ellos también ha influido el ajuste presupuestario, por lo que han considerado necesario reducir el gasto de papel en sus exámenes.

Aprovechando tal situación se han propuesto demostrar a sus alumnos la utilidad práctica de la asignatura por lo que han decidido cambiar el tipo de examen que sistemáticamente repetían cada año.

En el examen se proponen N problemas. Cada problema tiene un valor de v_i y ocupa un número de páginas p_i . Se trata de maximizar el valor de los problemas resueltos teniendo en cuenta que sólo pueden rellenarse P páginas. Ahora bien, puesto que consideran que resolver más de una cantidad dada de problemas debe primarse de algún modo, han añadido que si se resuelven M problemas pueden emplearse P' páginas más. Además, sólo tienen en cuenta los problemas completamente resueltos.

Resuelva el problema utilizando **Programación Dinámica**: Plantee la ecuación recurrente óptima y desarrolle el algoritmo que resuelve el problema.

Aplique el algoritmo desarrollado a la siguiente situación:

Se proponen 5 problemas con puntuaciones 90, 15, 40, 30 y 10 y que ocupan resueltos 90, 15, 45, 35 y 15 unidades de medida, respectivamente. Disponiendo únicamente de 100 unidades de medida, aunque si se realizan tres problemas se pueden utilizar 110 unidades de medida.

- 31.– **Dentro del Laberinto.** Representamos un laberinto cuadrado con una matriz booleana de N Filas por N Columnas. En el laberinto hay obstáculos. Desde una casilla se puede acceder a las adyacentes dentro de la misma fila o columna, nunca en diagonal, siempre y cuando en la casilla destino, no exista ningún obstáculo.

Partiendo de una casilla (i,j) , construya un algoritmo que nos devuelva el **camino mínimo (aquel que tiene el menor número de casillas) que debemos recorrer para alcanzar cualquiera de las salidas del laberinto (puede haber varias)**. La complejidad del algoritmo debe ser polinómica con respecto a N , y debe darse una estimación de la misma.

- 32.– De una relación de N problemas se van a extraer los P problemas de que consta un examen. Los M alumnos que van a concurrir al examen han entregado resueltos los N problemas y los profesores de la asignatura los han calificado. Cada problema se ha calificado con una nota entre 0 y 10. Si un problema no se ha entregado se ha calificado con 0. Tenemos pues la nota c_{ap} del alumno a en el problema p , para cada alumno a y para cada problema p .

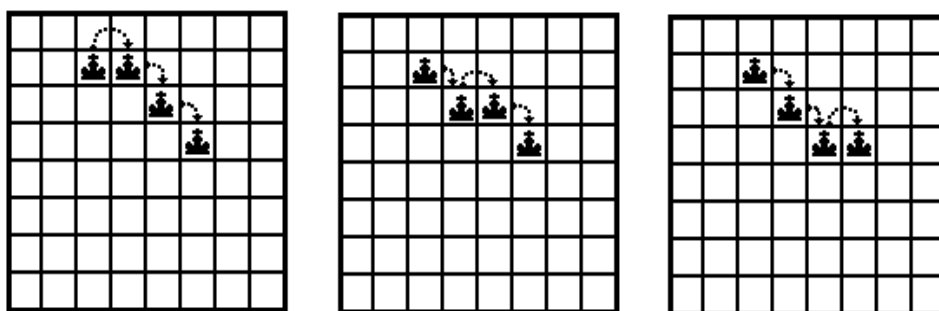
Diseñe un algoritmo basado en “**Programación Dinámica**”, que devuelva los P problemas que deben componer el examen para **maximizar el número de aprobados**.

- 33.– Una sala ha de estar constantemente iluminada por un mínimo de M bombillas. Para ello, disponemos de un conjunto finito de N bombillas $\{b_i\}$ que una vez encendidas no pueden apagarse y que se gastan al cabo de t_i días. Se trata de maximizar el número de días que podemos tener la sala iluminada.

Resuelva el problema usando: a) *Backtracking* y b) *Programación Dinámica*.

- 34.– Construya un procedimiento que nos devuelva el recorrido mínimo que ha de efectuar el rey negro para atacar todas las casillas del tablero de ajedrez partiendo de la casilla inferior izquierda del tablero.

- 35.– **Recorridos mínimos de un rey en un tablero de ajedrez.** Sobre un tablero de ajedrez tenemos únicamente el rey negro y deseamos trasladarlo de una casilla (i,j) a otra (m,n) , visitando el mínimo número de casillas posibles. Generalmente el problema no tiene una única solución, como por ejemplo puede observarse en la siguiente figura que nos muestra las posibilidades que tenemos de pasar de la casilla (2,3) a la casilla (4,6) por secuencias con mínimo número de pasos:



Construya un algoritmo que nos devuelva todas las secuencias mínimas de pasos por las que podemos trasladar el rey desde una casilla dada (i,j) a otra dada (m,n) . La complejidad de la solución obtenida debe ser $O(n)$, siendo n el número de secuencias mínimas de pasos.

NOTA: Para la obtención de una secuencia mínima partiendo de una casilla dada, hemos de dirigirnos a otra casilla que reduzca la "**distancia**" con la casilla destino.

- 36.– **Camino mínimo de un caballo para llegar de una casilla a otra del tablero.** Dadas dos casillas del tablero de ajedrez, **Origen** y **Destino**, construya un algoritmo que nos devuelva el camino mínimo que debe recorrer un caballo de ajedrez para que partiendo de la casilla **Origen** llegue a la casilla **Destino**. Si no existe ningún camino entre ambas casillas, el algoritmo debe advertirlo.

- 37.– **Caminos mínimos de un caballo para llegar de una casilla a otra del tablero.** Igual que el problema anterior, pero deben generarse todos los caminos mínimos entre las dos casillas. La

complejidad del algoritmo que nos genera un camino mínimo debe ser lineal con respecto al número de casillas que componen el camino. ¿La complejidad del algoritmo es lineal con respecto al número de caminos mínimos existentes entre las dos casillas?

- 38.– Se considera un tablero de ajedrez, un conjunto C de casillas del mismo, y dos casillas, *origen* y *destino*. Diseñe un algoritmo que devuelva el **camino mínimo del caballo para ir de la casilla origen a la casilla destino sin pasar por casillas del conjunto C** de casillas dado. NOTA: Se primará la eficiencia.

- 39.– Diseñe e implemente un algoritmo que devuelva una de las posibles soluciones al problema de "***La vuelta del caballo***", siempre que sea posible, para un tablero de tamaño n .

- 40.– **Recorrido del Caballo de Ajedrez con Condiciones.** Se considera la **lista ordenada** formada por **cuatro** casillas del tablero de ajedrez. Se trata de encontrar el camino que debe seguir el caballo para realizar la "***vuelta***" al tablero de tal forma que la primera casilla del camino sea la primera de la lista, y las restantes casillas de la lista se encuentran en el camino en el mismo orden relativo que en la lista, si bien no necesariamente son contiguas en el camino. Dicho de otra forma, dada la lista ordenada $\{c_0, c_1, c_2, c_3\}$ de casillas del tablero de ajedrez, hay que **determinar el camino que sigue el caballo en su "*vuelta*" al tablero tal que c_0 es su casilla inicial y la lista dada es una subsucesión del mismo.**

Diseñe un algoritmo que resuelva el problema para el caso de un tablero de dimensión N .

Considere la resolución del problema para el caso en el que únicamente sean **tres** casillas las que forman la lista considerada, una de las cuales es la casilla de partida.

En ambos casos, si el problema no tiene solución, el algoritmo desarrollado debe indicarlo.

NOTA: El problema de la "***vuelta***" al tablero del caballo de ajedrez consiste en que partiendo de una casilla origen recorrer todas las casillas del tablero visitando cada casilla una única vez y regresar a la casilla origen una vez visitadas todas las casillas del tablero, esto es, la casilla origen es accesible por el caballo desde la última casilla del recorrido.

- 41.– En Monilandia no conocen los billetes por lo que tan solo usan monedas; de modo que sus habitantes suelen ir encorvados por el peso de sus bolsillos. Por ello, se exige a los tenderos del país que a la hora de devolver el cambio a un cliente, se le entregue el menor número de monedas posible. Resuelva el problema utilizando:

i) ***Programación Dinámica***,

ii) ***Backtracking***.

Ciertamente en nuestro país el problema admite una solución más simple y eficiente. Exponga la condición necesaria para poder aplicar esta segunda política y programe dicha política, demostrando su optimalidad.

42.– En un país se emiten sellos de K valores distintos, y se dispone de sobres en los que caben un máximo de M sellos. Se trata de calcular las tarifas postales que pueden establecerse. Resuelva el problema:

i) Con **Programación Dinámica** (y calcule su complejidad).

ii) Con **Backtracking**.

43.– Suponga ahora que el empleado de correos es un maniático, y le gusta que la tarifa empiece y termine con un sello de la misma clase. Diseñe un algoritmo que nos devuelva todas las tarifas postales que dicho empleado admite.

44.– Desarrolle ahora un algoritmo que calcule todas las tarifas postales que pueden establecerse sin sellos repetidos, de tal forma que se minimice el número de sellos de la misma.

45.– Tenemos dos jarras de capacidades 3 y 4 litros. Construya un algoritmo que devuelva la secuencia de operación para aislar 2 litros.

46.– Dada una sucesión $X = \langle x_1, x_2, \dots, x_m \rangle$, otra sucesión $Z = \langle z_1, z_2, \dots, z_k \rangle$ es una **subsucesión** de X ($k \leq m$) si existe una sucesión estrictamente creciente $\langle i_1, i_2, \dots, i_k \rangle$ de los índices de X tal que para todo $j = 1, 2, \dots, k$, tenemos que $x_{i_j} = z_j$.

Por ejemplo, $Z = \langle B, C, D, B \rangle$ es una subsucesión de $X = \langle A, B, C, B, D, A, B \rangle$ con la correspondiente secuencia de índices $\langle 2, 3, 5, 7 \rangle$.

Dadas dos sucesiones X e Y , decimos que una sucesión Z es una **subsucesión común** de X e Y , si Z es una subsucesión de ambas. Por ejemplo, si $X = \langle A, B, C, B, D, A, B \rangle$ e $Y = \langle B, D, C, A, B, A \rangle$, la sucesión $Z = \langle B, C, A \rangle$ es una subsucesión común de ambas. La sucesión $\langle B, C, A \rangle$ no es la subsucesión común más larga de X e Y , ya que tiene longitud 3 y la sucesión $\langle B, C, B, A \rangle$, que es común a X e Y , tiene longitud 4. La sucesión $\langle B, C, B, A \rangle$ es una de las subsucesiones comunes más largas de X e Y , al igual que la sucesión $\langle B, D, A, B \rangle$, ya que no hay subsucesiones comunes de longitud 5 o superior.

Utilizando **Programación Dinámica**, diseñe un algoritmo que dadas dos sucesiones determine la subsucesión de longitud máxima común a ambas.

47.– Se tienen n objetos que se desea ordenar según las relaciones $< e =$. Por ejemplo, existen 13 ordenaciones diferentes entre 3 objetos:

$A = B = C$ $A = B < C$ $A < B = C$ $A < B < C$ $A < C < B$
 $A = C < B$ $B < A = C$ $B < A < C$ $B < C < A$ $B = C < A$
 $C < A = B$ $C < A < B$ $C < B < A$

Diseñe un algoritmo de **Programación Dinámica** que determine en función de n el número de ordenaciones diferentes. El tiempo a emplear ha de ser $O(n^2)$ y el espacio $O(n)$.

- 48.– Dados N enteros distintos. Calcule el número de árboles distintos que se pueden formar con un entero en cada nodo, supuesto que etiquetamos cada nodo con uno de los enteros.
- 49.– Construya e implemente un algoritmo que decida si dos matrices de adyacencia representan o no el mismo grafo, salvo enumeración de sus vértices.
- 50.– Una línea de trenes que da servicio a las ciudades comprendidas entre dos, está servida por k tipos de trenes que parten cada hora (los trenes de un mismo tipo) de la estación de origen. Los trenes nunca se adelantan y se conoce el tiempo que tardan en llegar a cada estación, y la lista de estaciones en que se paran. Como no todo tren para en todas las estaciones, resulta que ciertos recorridos exigen trasbordos en estaciones intermedios. Se trata de conocida la hora de llegada a una estación de partida P , encontrar la combinación óptima para llegar a un destino Q . Resuelva el problema:
- a) Con **Programación Dinámica**.
 - b) Con **Backtracking**.
- 51.– Un **camino hamiltoniano** en un grafo no dirigido es un camino que usa cada vértice del grafo exactamente una vez. Construya un algoritmo que decida si un grafo dado tiene un camino hamiltoniano.
- 52.– Un **camino euleriano** en un grafo no dirigido es un camino que usa cada arco del grafo exactamente una vez. Construya un algoritmo que decida si un grafo dado tiene un camino euleriano y otro que en caso de que exista lo imprima. Hemos fragmentado el problema decidiendo primero si existe dicho camino puesto que se puede aplicar una estrategia devoradora al mismo con una complejidad "pequeña", sobre todo comparada con el que nos devuelve el camino.
- 53.– **El juego del Mastermind**. Programe el juego del Mastermind, de N colores con M posiciones, bajo las hipótesis:
- i) Los colores no pueden repetirse, y
 - ii) Los colores pueden repetirse.

En ambos planteamientos debe tenerse bien presente que la combinación que se propondrá en cada jugada será consistente con los resultados obtenidos en las anteriores.

- 54.– Dados seis números naturales entre 1 y 100, se trata de encontrar la expresión en la que pueden intervenir, una, ninguna o varias veces, los operadores +, -, * y / (división exacta) y un subconjunto de los seis números, éstos sin repetirse, que evaluada nos devuelva el número más cercano a otro número natural dado menor que 1000. Construya un procedimiento que nos resuelva el problema utilizando:

- a) **Programación Dinámica,**
- b) **Backtracking.**

- 55.– Tres misioneros y tres caníbales se encuentran en la orilla de un río. Han acordado que a todos les gustaría llegar a la otra orilla. Pero los misioneros no están seguros de qué otras cosas han acordado los caníbales. Por tanto los misioneros quieren arreglar el viaje a través del río de forma tal que el número de misioneros en cada lado del río nunca sea menor que el número de caníbales que están en el mismo lado. La única barca disponible sólo de dos plazas. ¿Cómo podrán atravesar el río sin que los misioneros corran el riesgo de ser comidos?

Construya un procedimiento que nos resuelva el problema para N misioneros y N caníbales.

- 56.– Pepe está prendado de una rubia de ojos verdes. Sabe que los colores preferidos de la rubia son:

Blanco, Negro, Amarillo, Verde, Azul y Marrón,

y se ha propuesto vestirse con todos ellos. Así, se pondrá:

Chaqueta, Corbata, Camisa, Pantalón, Calcetines y Zapatos.

De todos ellos tiene los colores antes citados, salvo de los **zapatos** que los tiene de color:

Negro, Blanco y Marrón.

Además, se pretende presentar a la mujer de sus sueños con un **ramo de rosas**. Cada ramo tendrá sólo rosas de un color, que pueden ser de color:

Rojo, Rosa o Amarillo.

Sabe además que no debe repetir los colores con las prendas de vestir, y que las rosas nunca deben ser del mismo color que los **calcetines**, aunque para una misma indumentaria puede llevar rosas de distinto color.

¿Cuántas y cuáles son las combinaciones distintas que puede formar?

57.– Paseábamos tranquilamente por la EXPO'92, un florido día de mayo perdiéndonos entre los demás visitantes, hasta que nos encontramos a un francés gritando: *Vive la France!!!*. Como quiera que no paraba y que como transcurría el tiempo se excitaba más, dándoselas también de listo, y dándole, no al Burdeos, sino al tinto de Valdepeñas, aprovechamos el encontrarnos en una terraza de baldosas de colores:

Blanco, Negro, Rojo, Amarillo, Verde y Azul,

y puesto que la bandera francesa es de color **Azul, Blanco y Rojo**, en esta secuencia, le propusimos que se situara en cualquiera de las casillas azules de los lados del embaldosado y permitiendo el acceso desde cada baldosa a las adyacentes vertical, horizontal o en diagonal, sin pisar dos veces la misma baldosa, contruyése la secuencia que nos da la cantidad máxima de banderas francesas consecutivas.

58.– **La huerta del tío Joaquín.**

El tío Joaquín tiene una huerta rectangular de 100 m de ancho dividida en zonas de longitud múltiplo de 100 m. Cada zona tienen una calidad (de 1ª a 5ª). Puede sembrar N productos de los cuáles se conoce el rendimiento por Ha, según la calidad del terreno. En cada temporada se hacen S siembras que posteriormente se recogen, y por otra parte, llueve ll veces, hiela h y hace buen tiempo bt veces. Se tiene que:

- i) Al helar se pierde la mitad del rendimiento de lo sembrado, y el terreno no sembrado pierde un grado de calidad.
- ii) Al llover crece un 20% del rendimiento, y al hacer sol un 15%.
- iii) Tres temporadas de sol o lluvia consecutivas arruinan la cosecha.
- iv) Dos temporadas de lluvia sin sol en medio sobre un terreno no sembrado mejoran la calidad un grado.

Se pide:

A) Implemente la huerta y las operaciones que modifican su estado.

B) Buscar el óptimo funcionamiento (alternativa de siembras, cosechas y sucesos atmosféricos) durante dos temporadas. Cada siembra puede ser de 1 a H hectáreas.

NOTA: La parte A) del problema se refiere a Estructuras de Datos, y la resolución de B) a los esquemas algorítmicos.

59.– El dueño de un Salón de Banquetes planea las celebraciones de toda la semana. Su mayor problema es disponer de suficientes manteles limpios. El buen señor sabe de antemano los banquetes concertados para dicha semana, y por tanto, ha podido calcular los manteles necesarios para cada día (la función f , $f(i)$ es el número de manteles necesarios el día i -ésimo, $i = 1, 2, \dots, 7$). Además, se ha informado de los precios y velocidad en la entrega de las dos lavanderías más cercanas:

L_1 - x pts. por mantel, listos al día siguiente,

L_2 - y pts. por mantel, limpios en dos días, (un día más que L_1 pero con $y < x$).

Por otra parte, los manteles nuevos cuestan z pts. ($x, y < z$).

Ayude al señor a decidir:

A) La cantidad S de manteles nuevos que debe comprar al principio (se supone que el negocio es nuevo y todavía no dispone de manteles; además, la compra se hace el primer día y no se vuelven a comprar más).

B) El número de manteles que cada día de la semana se mandan a la lavandería (cada día se mandan a lavar todos los manteles ensuciados).

Por supuesto estas decisiones deben minimizar el gasto total en toda la semana.

60.- "*El nombre de la rosa*" o lo que no sabía Guillermo de Baskerville cuando visitó el laberinto.

Guillermo recordaba vagamente un procedimiento para recorrer el laberinto, pero no le daba gran fiabilidad ya que "probablemente no funciona en algunos casos". El algoritmo era el siguiente:

- Partimos de la escalera de la que parte un único pasillo.
- Al llegar a una habitación por primera vez, marcamos el pasillo por el que accedemos a ella.
- Al salir de una habitación inspeccionamos primero todas sus salidas, buscando si alguna nos lleva a una habitación nueva. Si encontramos una tal, pasamos a ella, y si no es así, retrocedemos por el camino de acceso marcado.
- Una vez en la escalera habremos recorrido el laberinto.

Se pide:

- a) Un procedimiento que recorra el laberinto y nos describa el mismo numerando las habitaciones la primera vez que se accede a ellas, y cite tal enumeración cuando las volvamos a cruzar. Para distinguir los pasillos supondremos que poseemos una perfecta orientación numerando los pasillos en una habitación según se recorran, en el orden indicado por el sentido de las agujas del reloj. Los pasillos se cuentan tantas veces como se recorren.

Pero Guillermo deseaba explorar muchas veces el laberinto, y, para ello, lo mejor sería que averiguase primero el recorrido óptimo: minimizando el número de pasillos atravesados.

- b) Encuentre tal recorrido con un algoritmo de Backtracking.
- c) Programe dicho algoritmo utilizando Backtracking.

NOTA: El laberinto es desconocido y, por tanto, aunque puede representarse en el ordenador como se desee, debe tenerse cuidado de no utilizar dicha información salvo en la medida en que nuestro recorrido nos lo suministra.

EJEMPLO de INCORRECTO uso de la implementación:

Laberinto es la matriz de adyacencia del grafo de pasillos, y se usa su tamaño para saber cuando se ha visitado todo el laberinto; o equivalentemente, se utiliza información asociada a habitaciones no visitadas.

- 61.– Cuenta la leyenda que en la Cámara Funeraria de la pirámide del faraón Ammenotitis XIV se sepultaron gran parte de las riquezas que logró reunir en su corta vida, un tesoro de casi incalculable valor. Si bien se pensó durante largo tiempo que la leyenda era una fábula y no tenía realidad, una expedición española al frente de la cual figura el famoso arqueólogo don Luis de Rigobares Langredo ha logrado descubrir el emplazamiento exacto de la pirámide, oculta a los ojos del mundo por la arena del desierto.

También ha descubierto que tiene **varias entradas**, y por las inscripciones (maldiciones) que rezan en las entradas de la gran pirámide ha deducido **que el interior es un laberinto de cámaras**, alguna de las cuáles son **pozos sin fondo**, comunicadas entre ellas por **galerías** entre las que hay **algunas que están obstruidas por escombros** caídos de las paredes fruto de un terremoto o no se sabe bien que fuerzas extrañas. Ha comprobado que **en atravesar una galería sin escombros tarda T minutos**, en tanto que **para una galería de la que ha de quitar los escombros tarda M veces más**.

Don Luís de Rigobares es un arqueólogo muy metódico y en sus exploraciones siempre que puede aplica un algoritmo **Backtracking (Vuelta Atrás)** apropiado.

- a) Construya un algoritmo que simule el recorrido del gran arqueólogo Rigobares en busca de la cámara funeraria del faraón, y nos diga cuál es la secuencia de cámaras que hemos de visitar sin recorridos inútiles hasta alcanzar el tesoro y el tiempo total que hemos tardado en encontrarla.

NOTAS:

En un **pozo** pueden desembocar varias galerías, ahora bien no podremos tomar ninguna de ellas porque nos es imposible atravesar el **pozo**.

Adviértase también que los escombros de una galería sólo se quitan la primera vez que se atraviesa la galería.

Algo que habrá que tener bien presente es que el laberinto se ha de representar de alguna forma en el ordenador pero que dicha información no podrá ser utilizada más que según la vayamos obteniendo al recorrer el laberinto.

- b) Suponga ahora que el malvado Malandrín, jefe de los ladrones de tumbas, se ha confabulado con sus secuaces para robar el sarcófago del faraón pero ha de hacerlo en el mínimo tiempo posible no para evitar la vigilancia de los guardianes sino porque una de los descubrimientos del insigne don Luís es el descifrar una inscripción en la cámara que advierte que tras retirar el sarcófago del lugar destinado al reposo del faraón la pirámide se derrumbaría atrapando a los usurpadores de la paz de Ammenotitis en pocos minutos.

Nótese que lo que se ha de minimizar es el tiempo que se tarda en salir al exterior partiendo de la cámara funeraria, y que hay varias posibles salidas (las entradas descubiertas por el gran Rigobares).

62.– Construya un algoritmo capaz de transformar un entero inicial cualquiera N en otro entero final M , con el menor número de transformaciones de la clase:

$$a) f(i) = 3 * i$$

$$b) g(i) = \text{Parte entera de } i/2.$$

Por ejemplo, para transformar 15 en 5 podemos hacer:

$$15 \rightarrow 7 \rightarrow 21 \rightarrow 10 \rightarrow 5, \quad \text{haciendo } ggfg.$$

Si la transformación es imposible el algoritmo debe detectarlo sin ciclar indefinidamente.

63.– Se considera el problema de Programación Lineal Entera: por medio de un algoritmo de **Backtracking (Vuelta Atrás)**.

$$\max \sum_{i=1}^n b_i x_i$$

sujeto a:

$$\sum_{i=1}^n c_i x_i \leq d$$

$$x_i \geq 0, x_i \text{ entero } i = 1, \dots, n,$$

siendo $b_i, c_i > 0$.

Diseñe un algoritmo lo resuelva basado en:

a) **Programación Dinámica**, o

b) **Backtracking**.

Aplíquelo al problema siguiente:

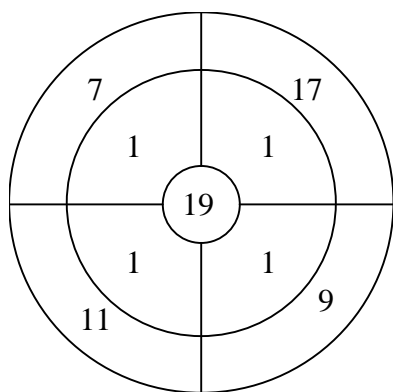
$$\max 4x + 3y + 2z$$

sujeto a:

$$3x + 4y + 2z \leq 8$$

$$x, y, z \geq 0, \text{ enteros}$$

64.– Dada la siguiente diana:



Diseñe un algoritmo que devuelva cómo sumar exactamente 100 puntos con el menor número de dardos. Resuelva el problema utilizando: i) **Programación Dinámica** y ii) **Backtracking**.

- 65.— Se considera una tabla de rendimientos R de dimensiones $M \times N$, donde R_{ei} indica el rendimiento obtenido al invertir en la entidad e una cantidad i . Diseñe un algoritmo basado en **Programación Dinámica** que maximice el rendimiento al invertir una cantidad C , entre las M entidades financieras de tal forma que se maximice el beneficio obtenido. NOTA: la inversión en una entidad se considerará global, esto es, no se puede invertir 2 y 3, en una misma entidad, puesto que a los efectos consideraremos el valor de 5 como inversión.

Aplíquese a la siguiente tabla de rentabilidades, suponiendo que dispongamos de 4 millones.

	Valor de la Inversión en Millones				
	0	1	2	3	4
Entidad 1	0	2	5	6	7
Entidad 2	0	1	3	6	7
Entidad 3	0	1	4	5	8
Entidad 4	0	0	2	5	9

- 66.— Tenemos un tablero de $M \times N$ casillas, en cada una de las cuales hay una cantidad de monedas, no necesariamente la misma cantidad en cada casilla, y además algunas casillas están vacías. Situados en una casilla, sólo podemos desplazarnos a una de las casillas adyacentes tanto en diagonal, horizontal como verticalmente, que tenga monedas, y cuando lo hacemos tomamos todas las monedas que hay en la misma.

Construya un algoritmo **Backtracking**, que devuelva el **recorrido mínimo** (el de menor longitud) necesario para reunir un número dado D de monedas, partiendo de una casilla dada.

- 67.— Tenemos N asignaturas optativas en la carrera, cada una con K_i créditos y un cierto horario H_i . Se trata de mostrar todas las posibles elecciones de asignaturas que podemos hacer para conseguir T créditos exactamente, de forma que dos asignaturas nunca coincidan en el horario y no se sobrepasen un número máximo M de asignaturas ($M < N$).

Resuelva el problema anterior suponiendo además que horario debe tener el menor número de asignaturas. En el caso de que haya varios, bastará con que muestre el primero

68.– **Partida de Dominó.**

Dado un reparto de fichas de dominó, construya una de las partidas más cortas en las que el jugador que sale gane, si es que existe alguna para dicho reparto.

69.– El juego del dominó tiene 28 fichas distintas. Dada una lista de las mismas halle:

- a) Todas las posibles **cadena correcta**s que se pueden formar con fichas de tal lista.
- b) Una de las **cadena correcta**s más largas (puede haber varias) que puede formarse con fichas de la lista.

NOTA: Una **cadena es correcta** si los extremos adyacentes de dos fichas consecutivas son del mismo número.

70.– **Cubrimiento de Conjuntos.** Sea $S = \{S_i\}_{i \in I}$, una colección finita de conjuntos finitos. Obtégase un recubrimiento mínimo de la misma.

Un **recubrimiento** de S es subcolección de la misma $S' = \{S'_j\}_{j \in J}$, (i.e. $\forall j \in J \exists i \in I, S'_j = S_i$), tal que todo conjunto de S esté contenido en la unión de los conjuntos de S' .

Un **recubrimiento** S' de S se dice **mínimo** si el cardinal de S' es mínimo entre todos los recubrimientos de S , es decir, el número de conjuntos de la subcolección S' es mínimo.

Por ejemplo: Sea $S = \{\{1,3,4\}, \{2,4,6\}, \{2,4\}, \{1,3\}, \{5,7\}, \{4,6\}\}$. Recubrimientos del mismo son:

$$S' = \{\{1,3,4\}, \{2,4\}, \{5,7\}, \{4,6\}\},$$

$$S'' = \{\{1,3\}, \{2,4,6\}, \{5,7\}\}$$

$$S''' = \{\{1,3\}, \{2,4\}, \{5,7\}, \{4,6\}\}$$

Resuelva el problema mediante i) **Programación Dinámica** y ii) **Backtracking**.

71.– Para *Forrest Gump* la vida era como una caja de bombones. Cada caja tenía bombones de varias clases. *Forrest* frecuentaba una tienda que tenía muchas cajas distintas, y *Forrest* siempre deseó tener bombones de todas las clases, pero su paga no le alcanzaba. Por ello, tenía que comprar **el menor número de cajas** pues todas las cajas tenían el mismo precio. Resuelva el problema utilizando una estrategia de **Programación Dinámica** o **Backtracking**.72.– Se define el **Peso de un camino** como la suma de los pesos de sus vértices.

Construya un algoritmo de **Programación Dinámica**, en el que dados un grafo G dirigido y con pesos en los vértices y un vértice v del mismo devuelva el camino, sin ciclos, de mayor peso de entre todos caminos con origen en v .

Nótese que un camino sin ciclos es aquel en el que no se repiten los vértices.

73.— Don Juan Tenorio y don Luís Megía.

Don Juan en tiempos mozos tenía por costumbre dormir por el día y cortejar por la noche a las damas que el destino a tiro le ponía. En una de sus correrías se encontró con su gran competidor don Luís Megía, y se retaron a visitar la mayor cantidad de damas que pudieran en una noche, teniendo en cuenta que deberían ganar de estas una prenda como prueba de haber logrado su objetivo. Con ello ambos sabían que deberían estar con la dama un cierto tiempo para conseguir la prenda. Como expertos en el arte sabían fielmente el tiempo necesario para conseguir la prenda prefijada.

Don Juan le ofreció a don Luís que fuese el primero en cortejar, descansando por tanto una noche, durante la que trabajaría Megía sin tener que competir con Tenorio, y a la noche siguiente sería don Luís quien descansaría y dejaría trabajar a don Juan. A la tercera noche ambos se reunirían y contarían las prendas ganadas.

Para mayor equidad han elegido una lista de las damas a cortejar que será para ambos igual. Ambos son conscientes de que gastarán también tiempo en sus desplazamientos, aunque en un acto de vanidad, estiman será mayor en proporción que el que tardarán en ganar la prenda de la dama.

Don Luís creyendo una ventaja el salir primero, accedió de buena gana a la propuesta de Tenorio. Craso error el suyo porque no estimó que don Juan contaría con la ayuda que le proporciona su escudero, fiel servidor de antaño, que estudió Programación y le ha servido en bandeja a Tenorio el recorrido óptimo que debe hacer esta noche para vencer a Megía. ¿Cuál es el recorrido que hará don Juan?

74.— En un campo tenemos un número determinado de flores, cada una con una determinada cantidad de polen. Tenemos un zángano que guiado por la ley del mínimo esfuerzo, sólo vuela a las flores adyacentes. Supuesto que debe saciar su apetito con una cantidad de polen P , visitando el menor número de flores, diseñe un algoritmo basado en la Programación dinámica que le ayude a conseguir su objetivo. Supóngase que parte de una flor dada.

¿Y si el zángano puede elegir la primera flor? Si quitamos la restricción de la elección a las flores adyacentes, ¿cuál sería la solución?

Calcule la complejidad del algoritmo desarrollado en cada uno de los casos.

75.— En un país netamente vitivinícola, cada uno de los N pueblos principales se ha especializado en la crianza de cada uno de los T tipos de vinos propios del país ($T \leq N$). Un extranjero por diversas circunstancias ha sido conminado por las autoridades a que abandone el país en el

menor tiempo posible. Este sujeto desea llevarse P tipos distintos de vino dados ($P \leq T$), que son los que le gustan especialmente. Dispone de un mapa del país con la distancia entre pueblos y el tiempo necesario para ir de uno a otro.

Sabiendo que se encuentra en una ciudad dada, construya un algoritmo que indique el tiempo mínimo que necesita para recoger los P tipos de vino y abandonar el país, y devuelva también el camino que seguiría. Resuelva el problema utilizando: i) **Programación Dinámica** y ii) **Backtracking**.

Modifique el algoritmo si basta con llevarse P tipos cualesquiera de vino. ($P \leq T$)

Suponga ahora que el ciudadano extranjero desea abandonar el país cuanto antes, luego no se detendrá a comprar vino, pero tiene que decidir el mismo cuál es el camino que debe seguir, y consecuentemente tiene que hacerlo en el menor tiempo posible. ¿Cuál es el algoritmo que habrá empleado para decidir el camino a seguir para abandonar el país y cuál es el itinerario que seguirá? Calcule la complejidad del algoritmo.

76.– Un sujeto ha tenido que hacer una escala en el aeropuerto de su ciudad natal y dispone de H horas hasta que salga su avión. Ha pensado en visitar a la mayor cantidad posible de parientes y amigos en sus casas de la ciudad. Para ello, conoce la distancia en tiempo que hay entre dos cualesquiera de las casas a visitar y el tiempo que debe estar en cada casa para que se considere una visita. También sabe el tiempo para ir desde el aeropuerto hasta cada casa y el necesario para ir desde cada casa al aeropuerto. Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que devuelva el recorrido que debe hacer para conseguir su objetivo.

77.– “Epe”, “El Programador Enmascarado”, es el nuevo héroe de los videojuegos. Ha llegado de incógnito a la ciudad y se ha alojado en un hotel. Tiene que “neutralizar” a M cualesquiera de los N “malvados” que planean un complot para sembrar el caos en la ciudad, con ello desbarataría el complot. Lo hará al abrigo de las sombras de la noche, mientras los “malvados” duermen en sus casas, y regresando a su hotel (para no levantar sospechas), en el **menor tiempo posible**. Sabe la distancia en tiempo entre cada casa, el tiempo que separa cada casa de su hotel y el tiempo que tarda en “neutralizar” a cada *malvado*. Diseñe un algoritmo que resuelva el problema utilizando una estrategia basada en **Programación Dinámica** o **Backtracking**. **NOTA:** Considérese que el tiempo cuenta desde que sale de su hotel y termina cuando regresa al mismo.

78.– Diseñe un algoritmo que resuelva el **Juego del Sudoku**.

79.– “El Luis” es un *vampiro* de la leche, porque forzado por las circunstancias –proximidad de los exámenes– se ha visto obligado a cambiar su dieta habitual, por dos litros de leche diarios, aunque el fin de semana vuelve a sus costumbres, e incluso también se permite algún que otro *lingotazo*.

La tensión ante los exámenes le ha llevado a plantearse, seriamente, cambiar sus estudios de Informática por maquinista de trenes, pero ya que le quedan pocas asignaturas piensa obtener el título (aún contando con la dificultad añadida por ciertos profesores en los exámenes de sus asignaturas).

En las últimas fechas la tensión se ha dejado sentir más, porque también necesita graduarse como *vampiro*, en otras palabras necesita víctimas a las que chupar la sangre. Y con la llegada del buen tiempo se le despierta el apetito: empieza a odiar la leche; y cada vez que ve algo rojo (un simple bolígrafo o rotulador) piensa en devorarlo –no se sabe bien si al objeto o al portador del mismo–.

Así pues, se ha planteado “matar varios pájaros de un tiro”: obtener el título, graduarse como vampiro, y dar rienda suelta a sus instintos, y todo esto, aplicando los conocimientos aprendidos en sus estudios de la carrera.

Se ha provisto de un mapa de las casas de la ciudad, ha medido en tiempo la distancia entre las mismas; sabe también el tiempo que tarda en “chuparle la sangre” a cada una de sus posibles víctimas; y sobretodo, tiene bien presente que ciertos “*sujetos*” deben **necesariamente** encontrarse entre sus víctimas.

Aunque tiene un problema: *la Unidad Antivampiros de la policía local*. Dicha unidad es capaz de localizar instantáneamente a todos los vampiros que no se encuentran en lugar seguro una vez que se ha disparado la alarma y darles caza. Así, “*El Luis*” debe estar en lugar seguro (su casa) antes de que se active la alarma.

Estudiando los métodos de actuación de la *Unidad Antivampiros* ha llegado a deducir que la alarma se dispara transcurridas T unidades de tiempo tras cometerse el primer crimen de la noche.

Diseñe la estrategia que empleará “*El Luis*” para, permaneciendo a salvo de la *Unidad Antivampiros*, lograr **maximizar el número de víctimas**, entre las que se encontrarán necesariamente los sujetos a los que les tiene especial “cariño” (un conjunto prefijado de las víctimas), con los siguientes supuestos:

- i) Una única víctima por casa.
- ii) “*El Luis*” chupa la sangre a todos los moradores de la casa, teniendo presente que lo que ahora tiene “*El Luis*” es el tiempo total que tarda en cometer su crimen en la casa, y no individualizado por víctimas.

Resuelva el problema por **Programación Dinámica** y por **Backtracking**.

80.– En la ciudad ha habido una invasión de *vampiros*, motivo por el cuál *La Autoridad* ha acordado permitir que en el jardín de cada casa (e incluso en los edificios de la Universidad) se puedan plantar ajos para espantar a los “*malignos*”.

Así, en la reciente construcción de una nueva urbanización, financiada por el ayuntamiento, se ha tenido en cuenta plantar en la parcela de cada casa ajos que protegerán a sus habitantes del ataque de los vampiros. Aunque tal medida tiene como contrapartida un determinado olor a ajo en cada casa.

Las casas que cuentan con financiación del ayuntamiento deben ser asignadas a las familias que las han solicitado. Para evitar quejas (e incluso abandono de la casas de la familia a la que se ha asignado) **se debe cumplir que el olor a ajo de la casa debe ser soportado por la familia que la habite**. Para cada familia tenemos el grado de tolerancia al olor a ajo.

Supuesto que conocemos el número de componentes de la unidad familiar (los animales de compañía quedan excluidos –entre éstos no se encuentra la suegra–), pretendemos **maximizar el número de personas protegidas** (cada casa es ocupada por una única familia), teniendo siempre presente que la familia que habite una casa debe soportar el olor a ajos de la misma. Desarrolle algoritmos que determinen la asignación utilizando estrategias basadas en **Programación Dinámica y Backtracking**.

- 81.– Tenemos un conjunto de n componentes electrónicas para colocar en n posiciones sobre una placa. Se nos dan dos matrices cuadradas C y D , de orden n , donde C_{ij} indica el número de conexiones necesarias entre la componente i y la componente j , y D_{pq} indica la distancia sobre la placa entre la posición p y la posición q (ambas matrices son simétricas y con diagonales nulas). Un cableado (x_1, \dots, x_n) de la placa consiste en la colocación de cada componente i en una posición x_i . La longitud total de este cableado viene dado por la fórmula:

$$\sum_{i < j} C_{ij} D_{x_i x_j}$$

Diseñe un algoritmo **Backtracking** que devuelva el **cableado de longitud mínima**.

- 82.– Juan Gorrilla, el aparcacoches, quiere trabajar menos y cobrar más (como cualquier hijo de vecino). Su trabajo le da la oportunidad de ganar mucho dinero gracias a las propinas que le dan sus “generosos” clientes. El problema es que está harto de tener que mover siempre los coches para poder conseguir aparcar un coche más (y por tanto, ganar una propina más). Juan tiene a su cargo una calle completa (con dos aceras) de L metros para poder aparcar los coches y cuenta con un conjunto de N coches a aparcar.

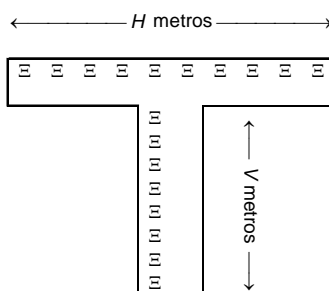
Diseñe un basado en **Programación Dinámica** o **Backtracking** que ayude a Juan a encontrar la **mejor disposición de los coches en las dos aceras de la calle**, sabiendo que cada coche tiene una longitud diferente y cada dueño de coche da una propina distinta, de manera que **Juan gane la mayor cantidad de dinero con las propinas aunque no consiga aparcar todos los coches**. Calcule la complejidad del algoritmo desarrollado.

Nota: Considere que en la longitud de cada vehículo, ya está incluida la distancia necesaria para poder aparcar el coche.

- 83.– En esta ocasión, Juan Gorrilla tiene que trabajar en una exposición de coches. Para los organizadores, es crucial que todos los coches que se aparquen de la misma marca (no es necesario aparcarlos todos) tienen que estar en la misma acera (aunque no es necesario que estén aparcados consecutivos). Los organizadores saben, a priori, que tendrán M marcas de

coches distintas para aparcar en las dos aceras. Resuelva el problema anterior bajo las condiciones anteriores.

- 84.– Juan, “*el gorrilla*”, ha sido contratado por el Casino DSAS3, sito en el número 13 de la Calle del Percebe (en donde, hasta hace poco, se erigía un antiguo bloque de pisos, habitado por excéntricos inquilinos). Su nuevo trabajo consiste en aparcar los coches de los clientes en el patio interior del inmueble. Dicho patio, reconvertido en parking de lujo, tiene forma de “T”, de forma que se pueden aparcar coches en batería en el lado horizontal de dicha “T” y en línea o cordón en el lado vertical de la “T”, que sirve además como pasillo de entrada al patio.



Ayude a Juan a encontrar la mejor forma de aparcar los coches, mediante un algoritmo basado en **Programación Dinámica** o **Backtracking**, de forma que consiga colocar la mayor cantidad de coches posibles y, de esta manera, pueda obtener una mayor suma de dinero en propinas.

Nota 1: En el patio solo cabe una hilera de coches en batería y otra a uno de los lados del pasillo de entrada.

Nota 2: Cada coche que llega al casino tiene unas dimensiones específicas $A_i \times L_i$, que ya incluyen el espacio de separación necesario para poder aparcarlo.

- 85.– **Vente p'Alemania Juan!** A pesar de no gustarle el fútbol, no ha podido sustraerse a las sugerencias que, “en cierto estado”, le ha hecho su buen conocido Vicente Bosch, y ha decidido acompañar a su pandilla a Alemania. “*El gorrilla*” una vez en Alemania ha visto que puede sacarse unos L euros trabajando de aparcacoches en los alrededores de los estadios en las horas previas a los partidos. Así, hasta se puede llegar a pagar el billete y la entrada – quiere acompañar a su peña a ver el partido, aunque no le gusta mucho el fútbol.

Ha realizado una inspección por los alrededores del estadio y ha visto que tiene dos aceras en las que puede realizar su trabajo. En una se puede aparcar en batería, y en la otra en cordón, de longitudes L_B y L_C , respectivamente. Además, va a proporcionar un servicio de vigilancia porque ha encontrado a un compatriota, Darío, dispuesto a “cuidar” de los coches el tiempo que estén aparcados mientras que tenga lugar el encuentro. Por aparcar y cuidar cada coche “*el gorrilla*” recibe una propina que es diferente según el coche. Además, Darío le ha sugerido que aparque los coches de la misma marca en la misma acera, y Juan no ha puesto ningún reparo y ha accedido. Como “*el gorrilla*” no tiene muy claro como aparcar los coches para maximizar el beneficio, ha pedido ayuda a su amigo “*el estudiante*”, que estudia Ampliación de Programación. Y éste le ha dicho que no tenga cuidado que se lo hace en un *pis-pas* por **Programación Dinámica** o **Backtracking** con su inseparable portátil (incluso se lo lleva al fútbol).

Póngase en lugar de “*el estudiante*” y diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que dada una lista de N coches, cada uno con sus dimensiones, propina y marca, devuelva la solución para obtener el **mayor beneficio, aparcando los coches en ambas aceras, en una en batería, y en la otra en cordón, y aparcando los coches de una misma marca en la misma acera.** **NOTA:** En las dimensiones de cada coche se hayan incluidas las distancias necesarias para aparcarlo.

- 86.– Juan “*el gorrilla*” se va una semana de vacaciones a Syldavia. Ha realizado varias consultas por la Web, y se ha informado de los monumentos y lugares de obligada visita. En realidad que debería visitar, porque es más que posible que dadas las restricciones de tiempo que tiene no pueda visitarlos todos. En lo que sigue hablaremos de lugares a visitar (N), entendiendo que un lugar puede ser una ciudad con varios monumentos, o también el nacimiento de un río, una cascada, un castillo, etc. Con este motivo, metódico él, ha realizado las oportunas averiguaciones y le ha asignado a cada lugar un “placer” al visitarlo, y el tiempo que tardará en visitarlo. Si en un lugar existen varios monumentos a visitar, se entiende que el tiempo y el placer son en visitarlos todos.

Como piensa aprovechar sus vacaciones al máximo, piensa en alquilar una autocaravana y llevarse con él a su cuadrilla, sus tres inseparables, “*el pinta*”, “*el jarra*” y “*el botilla*”. De tal forma que van a dormir en la autocaravana y así se ahorran pasta en alojamiento y transporte. Naturalmente, ha calculado meticulosamente el tiempo de desplazamiento entre cada dos lugares.

Supuesto que tiene varias opciones para comenzar su visita por el país, ya que hay M aeropuertos que tienen conexión directa con el de su ciudad, diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que determine cuál es la ruta a seguir para maximizar el placer en la semana de vacaciones de que dispone.

- 87.– Darío “*el presi*” y Juan “*el gorrilla*” tras comprobar por sí mismos en sus anteriores ocupaciones el partido que le pueden sacar a la Informática, han acabado matriculándose en la Escuela de Informática, y de entrada se han matriculado en Ampliación de Programación, y de libre elección Programación Concurrente. ¡Porque ellos ya tienen alguna experiencia con esas máquinas!.

Influenciados por el cine, la idea que tenían de la Universidad era de ir al bar, pasear por parques y jardines, quedar con alguna que otra colega, ir a fiestas, participar en otras actividades extraacadémicas... Y algún día se acercarían por clase para conocer al profesor, relacionarse con los compañeros (pensaban sacarle un buen partido a estas relaciones), y sobretodo aprovechar para contemplar las bonitas vistas que se ven desde la ventana de la clase...

Mala suerte la suya cuando han dado con sus huesos en las clases situadas en los sótanos del politécnico. Para Darío aquello eran las *Mazmorras del Politécnico*.

Esas clases del edificio le recordaban demasiado su anterior morada (y cosas peores). Ahora no había tanto metal, pero era mucho cemento, y seguía sin ver la luz. Juan, acostumbrado a tener el cielo por techo, también se esperaba otra cosa. Con estas condiciones, no se sorprendían de las rarezas de los informáticos que habían conocido.

Pero uno y otro habían captado las posibilidades que se les ofrecían. Darío en un sitio con poca luz, y mucho trájín de gente, se desenvuelve en su ambiente para realizar su trabajo, además con alguna que otra pequeña colaboración podría “ocasionalmente” producirse algún que otro apagón (se va un diferencial por sobrecarga,... o un dedo que se equivoca al bajar un interruptor). El ruido de ciertas máquinas también podría ser su aliado. No estaba él pensando en la dificultad en dar clase cuando tienes al otro lado de la pared un taller con máquinas funcionando, más bien, en la oportunidad de que los incautos extraviaran alguna de sus pertenencias...

Así que cuando le explicaron algunos algoritmos de optimización puso la mayor de las atenciones al ver la utilidad de los mismos. Lo primero que hizo fue asegurarse de la fiabilidad de los datos de entrada... al algoritmo, se entiende. Y también de las salidas,... del edificio. Se hizo con los planos de las clases, distancia en tiempo y espacio entre ellas y otros lugares del edificio (principalmente las posibles salidas), los horarios de ocupación de las clases, y la posible “rentabilidad” que le podría sacar al “trabajarse” a una clase, tanto a los alumnos como al profesor (llevaba más de un mes asistiendo a clase todos los días para conocerlos mejor), y el tiempo que le llevaría hacerlo. Además, salía y entraba continuamente de clase, por lo que nadie se extrañaba cuando llegaba tarde, o se marchaba antes de que terminase.

Por supuesto, no pensaba limitarse a las clases, también había otras dependencias que podían ofrecerle buenos frutos. Si bien “trabajaría” en ellas únicamente cuando no hubiera nadie. Como siempre, había obtenido por otras fuentes el “rendimiento” del trabajo y el tiempo que tardaría en hacerlo.

Era muy bueno en su trabajo y estaba muy seguro de que desde su primer trabajo tardarían un tiempo T en descubrir que se ha producido algún “extravío”.

Recogidos todos los datos se puso a programar un algoritmo para dado un día sacar el máximo rendimiento a su trabajo. Darío era muy bueno programando, diseñe uno de los algoritmos que posiblemente Darío programase si utilizara **Programación Dinámica** o **Backtracking**.

Supóngase que al trabajar en una dependencia realiza todo el trabajo, no sólo una parte, y además de un tirón (no comenzará un trabajo si no puede realizarlo sin interrupciones). Supóngase también que el rendimiento que se obtiene en una clase es independiente de la hora, con tal de que esté ocupada.

88.— Tenemos un rompecabezas de colores cuyas fichas son cuadrados de l cm (UN cm) de lado, con un color central y las aristas pintadas de colores. Se trata de encajarlas en un tablero cuadrado de M cm de lado, teniendo en cuenta que dos aristas en contacto han de tener el mismo color. Se supone que tenemos una cantidad fija N de fichas. Se pide:

- a) Construya un algoritmo que decida si puede resolverse el rompecabezas.
- b) Resuelva el rompecabezas utilizando el mínimo número de fichas rojas.

89.– María Cruci Grama es una gran aficionada a los crucigramas, y a la hora de reformar su salón se le ha ocurrido una forma original de embaldosarlo. Ha comprado N baldosas con distintas letras pintadas en ellas (una letra en cada baldosa) y pretende que su salón tenga **la mayor cantidad de palabras formadas en horizontal (de izquierda a derecha) o vertical (de arriba abajo)**. El salón de María tiene una superficie de $P \times Q$ baldosas.

Diseñe un algoritmo basado en **Programación Dinámica o Backtracking** que le ayude a embaldosar su salón obteniendo la **mayor cantidad de palabras**.

Notas:

- a) María dispone de un diccionario electrónico que mediante la llamada a la función **Boolean EsPalabra(String palabra)** puede saber si una combinación de letras forman una palabra.
- b) Se asume que una misma baldosa puede formar parte de varias palabras, incluso en la misma dirección.

90.– Dada una lista de P palabras, se trata de colocarlas en forma de crucigrama en una cuadrícula de $M \times N$ en la que hay C casillas negras (utilizadas para separar las palabras). Diseñe un algoritmo **Backtracking** que realice esta colocación o indique si esto no es posible. Resuelva el problema bajo los siguientes supuestos:

- a) las casillas negras ya se encuentran fijadas,
- b) el algoritmo debe poner él las casillas negras.

91.– Los M profesores de Ampliación de Programación de la Universidad de Informática de Tomelloso no tienen alumnos tan pelotas como los de la Escuela Superior de Informática, aunque son casi tan vagos como éstos.

Han realizado el examen final y de entre todos los que han suspendido N alumnos se han apuntado para la revisión. Sabemos el tiempo que tarda cada profesor en atender a cada alumno en la revisión, y que cada profesor revisará al menos un examen.

Las revisiones son el mismo día y empiezan a la misma hora. Los profesores desean terminar la revisión lo antes posible. Diseñe un algoritmo que resuelva el problema utilizando **Programación Dinámica o Backtracking**.

92.– ¡¡Una horda extraterrestre ha atacado la tierra indiscriminadamente bombardeando aquí y allá todas las ciudades que han encontrado a su paso!!

Por suerte el “ELE”, *Ejército de Liberación Europeo*, ha conseguido derribar N naves cayendo estas en N sitios distintos del globo terrestre.

Por suerte para nosotros, Mulder y Scully estaban por ahí realizando unas escuchas, cuando interceptaron un mensaje intergaláctico que avisaba de la invasión definitiva de la tierra en D días.

En este momento decidieron ir a M cualesquiera de los N sitios ($M < N$) donde habían caído las naves enemigas para analizar los restos de estos seres y poder fabricar un arma biológica que terminase definitivamente con ellos.

Diseñe un algoritmo basado en **Backtracking** o **Programación Dinámica** para ayudar a Mulder y Scully a visitar los M sitios necesarios, en el menor tiempo posible y siempre antes de que lleguen los extraterrestres, para recoger las muestras y preparar las armas biológicas.

Todo debe estar apunto para cuando lleguen “los hostiles”.

- 93.— Tenemos un conjunto de N hombres y N mujeres. Para cada uno de estos conjuntos tenemos sendas matrices cuadradas de $N \times N$ elementos, H y M , donde M_{mh} indica la preferencia de la mujer m sobre el hombre h , y análogamente con H para los hombres. Construya un algoritmo que encuentre el emparejamiento de hombres y mujeres que maximice la suma de los productos de las preferencias.
- 94.— El encargado de una red de concesionarios ha decidido rebajar el precio de una serie de N coches que no han sido vendidos y que se han pasado de moda. Los coches son voluminosos, con una longitud y un peso determinados. Para la distribución de los coches se utilizan M trailers. Cada trailer tiene una longitud dada, y puede soportar un peso determinado (no tienen porqué ser los mismos para todos). Se deben repartir la mayor cantidad posible de coches. Diseñe una estrategia basada en **Backtracking** o **Programación Dinámica** para resolver el problema.
- 95.— Disponemos de n cajas de longitud C_i y anchura M cada una. Se pretende empaquetar K objetos, de longitud O_j ($O_j > M$) y anchura M cada uno, utilizando el mínimo número de cajas. Diseñe un algoritmo **Backtracking**, que nos de la mejor disposición de los objetos en las cajas para conseguir el objetivo de utilizar el menor número de cajas. ¿Es óptimo? Calcule su complejidad.
- 96.— Tenemos N libros y K estantes de una estantería. Cada libro tiene un ancho A_i y peso P_i distinto y cada estante puede soportar un peso máximo PM_j . Todos los estantes tienen la misma longitud L . Diseñe un algoritmo utilizando **Backtracking** o **Programación Dinámica** que encuentre la disposición de los libros en las estanterías de manera que podamos colocar el mayor número de libros.
- 97.— **Viernes13.com**. Jason había decidido cambiar sus hábitos. Últimamente le habían estado comiendo el tarro con las bondades de la aplicación de la informática a problemas de optimización, y él que siempre estaba pensando en lo mismo, pensaba ir ahora por los cibercafés y otros garitos.com.

Para ello, se había provisto de un plano en el que se detallaba la distancia (en tiempo) t_{ij} entre los garitos.com, y también tenía una estimación del número de posibles víctimas de cada garito.com para esta noche v_i , porque Jasón sólo tenía esta noche para cometer sus fechorías. Y también sabe el tiempo que tardará en “liquidar” a todas las víctimas de cada garito t_i (distinto para cada garito).

Diseñe un algoritmo de *Programación Dinámica* o *Backtracking* que nos devuelva la secuencia de garitos.com que visitará Jasón, para “cazar” la mayor cantidad de víctimas a lo largo de toda la noche, suponiendo que puede empezar por uno cualquiera de los garitos.com.

98.– **Un repartidor de pizzas tiene que distribuir N encargos**, que le han asignado previamente, y tratará de hacerlo **en el menor tiempo posible**, porque desea “pegarle a la hebra” con la vecina del quinto. Para ello, se ha provisto de un mapa de la ciudad, en el que ha señalado las casas a las que tiene que llevar encargos (un encargo por casa), y en el que ha anotado la distancia, en tiempo, entre casas a las que tiene que llevar encargos, y también la distancia (en tiempo) de la central de reparto hasta cada casa. Supuesto que en cada viaje puede llevar como mucho M pizzas, diseñe un algoritmo que resuelva el problema utilizando una estrategia basada en *Programación Dinámica* o *Backtracking*.

99.– *El tío Eusebio* quiere rememorar sus tiempos mozos, y se ha propuesto hacer un bombo para mostrarle a sus nietos la técnica de construcción de los mismos. Además, lo hace con otro propósito que más adelante se verá.

Cuando era joven no tenía miedo a las inclemencias del tiempo y era capaz de seguir en plena faena cuando la más negras nubes amenazaban tormenta en pleno mes de siega, o ensarmentando aunque estuvieran cayendo chuzos de punta. Pero los años no perdonan, y ahora tiene bastantes achaques, aunque conserva su buen sentido para predecir el tiempo, el mismo que usa para dar los consejos a los jóvenes, que las más de las veces ignoran.

Tiene varias hazas y desea que el bombo le sirva para resguardarse de las inclemencias del tiempo, principalmente agua, porque nunca se le ocurriría meterse en el bombo cuando viniera una tormenta con aparato eléctrico, salvo que el bombo fuera “último modelo” dotado de pararrayos –había perdido a alguno de sus mejores amigos que no habían seguido el consejo–.

Principalmente tiene viñas, aunque también tiene alguna que otra siembra, y con la jubilación, tiene tiempo para visitarlas regularmente –casi a diario–, por lo que sabe la distancia en tiempo, entre ellas. Lo mide en tiempo porque se ha tenido que acostumbrar a las horas para tratar con los jóvenes, esos que se desesperan cuando hablan de “estar un rato”, pero a él nunca lo han tenido que esperar como él tiene que hacer con ellos, y eso que se pasan la vida mirando el reloj pero nunca llegan a tiempo...

Se trata de **hacer el bombo en el haza a la que se tarde lo menos posible en llegar a él desde cualquiera de las otras hazas**. El algoritmo desarrollado debe tener una **complejidad polinomial**, con respecto al número de hazas.

- 100.—Enterado *el hermano Paco* de las intenciones de su amigo *Eusebio*, quinto suyo, y acompañante de sus caminatas por el campo, le ha propuesto que en vez de uno hagan dos: “Porque no sólo vamos a tus tierras, sino también, y las más de las veces, a las mías, y así, con dos, mejor nos resguardaremos. O no te acuerdas del refriao que pillaste esta primavera, en *Las Chimeneas*, cuando fuimos a ver el centeno”. —Le ha dicho con el acento que le caracteriza.

La idea que les guía sigue siendo hacer los dos bombos en aquellas hazas que les permitan llegar a uno de ellos en el menor tiempo posible desde cualquiera de las otras hazas.

Ellos tenían ya elegido las hazas donde hacerlos, pero uno de sus nietos que estudia Informática, y que al principio creía que eso de los bombos eran como tiendas de campaña que se compran en las tiendas de aventura, les ha dicho que él les dirá el lugar donde hacer los bombos. Y han accedido. Pero se temen lo peor porque todavía se están riendo de cuando su nieto les preguntó que ¿cuánto cemento necesitarían para hacer un bombo?

- 101.— K atracadores consiguen robar n_1 billetes del tipo 1, n_2 billetes del tipo 2, ..., n_p billetes del tipo p . Quieren repartir el botín de forma que todos se lleven exactamente la misma cantidad de dinero, donando el resto a la asociación *ALELADOS* (Asociación de Ladrones Escasamente Listos y Atrapados). Naturalmente, se trata de donar a dicha asociación la menor cantidad de dinero posible. Construya algoritmos que resuelvan el problema según: i) **Programación Dinámica** y ii) **Backtracking**.

- 102.—Una empresa constructora de barcos fabrica en sus N astilleros M modelos de barcos, construyendo mensualmente b_{ak} barcos en cada astillero a de cada modelo k . La fabricación de estos barcos es de M_a millones por astillero y por mes.

Calcule como repartir la fabricación en los astilleros para reducir al mínimo el coste teniendo en cuenta que se debe hacer en un número T de meses, y se quieren construir como mínimo b_k barcos de cada modelo k .

- 103.—**Hace mucho tiempo, en una galaxia muy, muy lejana...**

La *Alianza Rebelde* ha obtenido información acerca del emplazamiento de varias bases imperiales en *Endor*. Deben destruirse todas las bases, pero nuestras naves de carga han sido interceptadas...

Nuestra única posibilidad es un bombardero *Y-Wing* cargado con T torpedos. Hay M tipos de torpedos, cada uno de los cuales tiene una potencia t_k . El bombardero lleva n_k torpedos de cada tipo. Cada base imperial debe ser bombardeada con una potencia P_d para ser destruida.

Dado que el número de bombas es limitado, diseñe un algoritmo que **minimice el número de bombas a usar, si puede destruir todas las bases, o en su defecto maximice el número de bases destruidas**.

Que la Fuerza te acompañe...

- 104.– *Caperucita Roja* ha sido enviada por su abuelita al bosque para recoger flores que inunden su casa con la mayor intensidad de fragancia posible. Dispone de N floreros, cada uno de los cuales puede llegar a contener un peso de flores P_i y un volumen V_i . Cada flor del bosque tiene un volumen v_j , un peso p_j y una intensidad de fragancia f_j . Como la primavera acaba de llegar la mayoría de las plantas aún no han brotado, por lo que en el bosque tan sólo hay F flores. Construya un algoritmo utilizando **Programación Dinámica** o **Backtracking** que ayude a *Caperucita* a seleccionar las flores y decidir en que florero las meterá para maximizar la cantidad de fragancia en la casa de su abuelita.

Una vez que *Caperucita* ha decidido las flores que tiene que recolectar, se le presenta el problema de recogerlas en el menor tiempo posible, puesto que desea irse a jugar con ese que llaman “*El Lobo Feroz*”, y tiene el problema que las flores se encuentran muy dispersas en un bosque tan grande. *Caperucita* que es muy previsora, en sus continuos paseos por el bosque ha ido anotando la localización de las flores, y sus distancias (en tiempo) entre ellas, así como la distancia a casa de su abuelita.

Puesto que la cesta de *Caperucita* tiene una capacidad en volumen V y que no quiere portar más de un peso P , para no estropear las flores. Calcule el tiempo mínimo para recolectar las flores seleccionadas en el apartado anterior.

- 105.– “*El Dedos*” ha pensado que lo de las cajas fuertes tiene más riesgo que asaltar sistemas informáticos, así que se ha puesto manos a la obra y se ha apuntado a un curso acelerado de Informática, eso sí, sin pagar por adelantado, porque casualmente había visitado jornadas antes de su clausura las oficinas de Opening y había visto lo que había (o mejor, lo que no había) en su caja fuerte.

Como ejercicio para iniciarse en el nuevo arte se había propuesto romper la protección del ordenador de “*El Profe*”. Y como buen observador que era, se había percatado que la contraseña que protegía el acceso al equipo tenía M caracteres, entre los que había L letras y el resto eran dígitos, y que además, no había N números seguidos.

Diseñe el algoritmo que debe programar “*El Dedos*” que genere todas las posibles contraseñas. Debe cuidarse la eficiencia.

- 106.– Darío, “*el presi*”, ya ha intentado escapar de la cárcel 54 veces, por el laberinto de alcantarillas que pueblan el subsuelo de la prisión. Siempre lo han pillado, pero en la última ocasión casi lo consigue, si no hubiera sido por un gato que maulló cuando no debía. Mañana piensa volver a intentarlo, y esta vez quiere que sea la definitiva.

No obstante, los guardias de la prisión piensan poner fin a esta situación y han contaminado el aire de unas cuantas habitaciones del laberinto con un gas somnífero, con el fin de evitar la fuga de Darío. Pero Darío no está dispuesto a quedarse a la sombra más tiempo y ha utilizado sus contactos para conseguir un mapa de las alcantarillas donde están marcadas las salas contaminadas, además, le han informado que podrá soportar la exposición al gas somnífero el tiempo equivalente a atravesar tres salas antes de quedarse dormido como un angelito.

Diseñe un algoritmo basado bien en **Backtracking** o bien en **Programación Dinámica** que ayude a Darío a encontrar la salida del laberinto por el camino más corto, sin pasar por más de tres salas contaminadas, de manera que la quincuagésima quinta vez sea la vencida y consiga la, tan ansiada, libertad.

107.–Darío, "el presi", ha sido atrapado otra vez en su ya quincuagésimo quinto intento. Nuevamente, piensa volver a intentarlo, y nuevamente los guardias de la prisión piensan poner fin a esta situación y han contaminado el aire de unas cuantas habitaciones del laberinto con un gas somnífero, en otras han puesto trampas, y en otras animales carnívoros, con el fin de evitar la fuga de Darío. Pero Darío no está dispuesto a quedarse a la sombra más tiempo y ha utilizado sus contactos para conseguir un mapa de las alcantarillas donde están marcadas las salas contaminadas, las trampas y los bichos. Antes de la fuga Darío cuenta con 100 puntos de vida. Teniendo en cuenta que atravesar una sala con gas le resta 20 puntos de vida, una con trampa 25 y otra con bichos 30. Diseñe un algoritmo utilizando **Backtracking** o **Programación Dinámica** que nos devuelva el camino de huida para Darío, tal que le quede el máximo de vida disponible para poder disfrutar de su libertad.

108.–Darío, "el presi", ha sido atrapado otra vez en su quincuagésimo sexto intento. Nuevamente, piensa volver a intentarlo, y nuevamente los guardias de la prisión piensan poner fin a esta situación. Esta vez los guardias de la prisión han soltado perros rabiosos por algunas salas del laberinto (los perros están entrenados para aguardar a Darío sin moverse de la sala). Suerte que un compañero de prisión de Darío, Filomeno "el carni", le ha facilitado un mapa con las salas donde puede encontrar chuletones de ternera para entretener a los perros. De esta forma, Darío podrá pasar por una sala donde haya un perro sólo si tiene un chuletón para echarle.

Diseñe un algoritmo **Programación Dinámica o Backtracking** que nos devuelva el camino de huida para Darío. Naturalmente, se trata de huir en el menor tiempo.

109.–Darío, "el presi", está harto de que le pillen cada vez que intenta escaparse de la cárcel y ha decidido que si tras el quincuagésimo-séptimo intento (en el que nada podía fallar, pero, inexplicablemente, algo falló...) no ha conseguido escaparse, nunca conseguirá burlar la vigilancia a la que es sometido por los agentes de la prisión de máxima seguridad.

Tras desistir de hallar la libertad por la vía rápida, Darío, se ha convertido en un preso modelo y va poner en marcha un nuevo plan para verse cuanto antes en la calle. Su idea consiste en apuntarse al plan de trabajos de reinserción que ofrecen en la prisión, para poder reducir cuanto antes, por buena conducta, los trescientos veinticuatro años, siete meses y un día que aún le quedan por pasar a la sombra.

En la prisión se ofrece una serie de n trabajos de reinserción de carácter social, que abarcan desde el macramé a la cría de gamusinos, pasando por el ikebana y el cultivo de tulipanes tornasolados. Pero a Darío le da igual lo que tenga que hacer en cada actividad, lo que realmente le interesa es que cada uno de esos trabajos le reducen la pena en un tiempo t_i . Cada uno de los trabajos tiene un horario semanal fijado de antemano, y Darío no piensa renunciar a ciertas actividades, su partidita de mus, comer, dormir y hacer vida social en el patio de la prisión.

Realice un algoritmo basado en **Backtracking** o **Programación Dinámica** que ayude a Darío a elegir el conjunto de trabajos que debe realizar para conseguir la mayor reducción de su pena, sabiendo que no puede realizar dos trabajos a la vez (los horarios de los trabajos pueden solaparse) y que una vez que se compromete a realizar uno de ellos debe realizarlo por completo para conseguir la reducción de la pena.

110.–Darío, "el Presi", después de fallar por sexagésima-sexta vez en su intento de fuga, ha sido obligado a trabajar como cartero dentro de la prisión. Su trabajo consiste en repartir la correspondencia a todos los reclusos de la prisión. Para Darío, esto es una contrariedad, ya que no le permite concentrarse en su *hobby* preferido, planear una nueva fuga. Por esto, ha decidido realizar su trabajo de forma correcta (para evitar represalias por parte del alcaide de la prisión), pero empleando el menor tiempo posible. Para planificar el reparto de hoy, Darío ha estado midiendo la distancia existente entre los distintos puntos de entrega de la correspondencia, y ha construido una matriz de distancias, con el tiempo de desplazamiento, y el tiempo empleado en entregar la correspondencia en cada punto. Darío cuenta con un carrito en el que puede llevar toda la correspondencia, y pretende repartir toda la correspondencia en el menor tiempo posible. Resuelva el problema utilizando un algoritmo basado en **Programación Dinámica** o **Backtracking**.

111.–Darío, "el Presi", después de fallar por sexagésima-séptima vez en su intento de fuga... sigue intentándolo. Quiere fugarse de la cárcel por el sistema de alcantarillado, pero tiene dos problemas: El diámetro de la bola que arrastra es de 50 cm y resulta ser demasiado grande para pasar por algunos pasillos. Y otro es que los pasillos que comunican unas alcantarillas con otras tienen demasiada pendiente y sólo puede circular por ellos en un sentido. Varias alcantarillas tienen su salida fuera de la prisión. Darío ha conseguido hacerse con los planos que le indican las salidas, el diámetro de los pasillos, el sentido de las pendientes y el tiempo para recorrer cada pasillo.

Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que ayude a Darío a encontrar el camino que le lleva hacia la libertad en el menor tiempo posible.

112.–Miguel "el estudiante", que el único título que tiene es la etiqueta de anís "El Mono" en la pared de su celda, es el mejor amigo de Darío "el Presi". Miguel ha terminado de cumplir condena. Se ha dedicado todos estos años a la gestión de la Biblioteca del centro penitenciario. Debido a su buena conducta y su amor por los libros se le ha concedido el deseo que tanto añoraba: tener su propia colección de L libros. Antes de marcharse tiene que decidir que libros quiere llevarse, para ello, se le dan C cajas. Cada caja i tiene unas dimensiones $L \times H \times A_i$, es decir, únicamente cambia el ancho de la caja ($A_i < H < L$). Además cada libro j tiene unas dimensiones $L \times H \times g_j$. (**Nota:** Los libros solo caben "de pie").

Como Miguel es un negado de la lógica y cuando salga de la cárcel quiere pasar el máximo tiempo posible leyendo, le pide a su amigo Darío "el presi" que le ayude para poder llevarse todas las cajas bien colmadas de libros. Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que le ayude a Miguel a llevarse la mayor cantidad de libros posible.

- 113.—Un sistema informático tiene que descargar un conjunto de N paquetes de actualización cada uno de los cuales tiene un tamaño Tam_i Kb. Para la descarga de esos N paquetes se dispone de S servidores remotos cada uno de los cuales cuenta con un ancho de banda de V_j Kb/s. El sistema de descarga solo permite descargar un paquete de cada servidor a la vez, pero podemos descargar varios paquetes a la vez procedentes de diversos servidores. Desarrolle un algoritmo que resuelva el problema propuesto indicando qué paquetes debemos descargar de cada servidor, de tal manera que se **minimice el tiempo de descarga**.
- 114.—Tenemos un procesador que debe realizar una serie de N trabajos. De cada trabajo i sabemos el instante exacto c_i en que debe comenzar, su duración d_i y el beneficio b_i que proporciona si se realiza. El procesador solo puede realizar una tarea al mismo tiempo. Supuesto que el procesador está operativo entre los instantes C y F , diseñe un algoritmo basado en **Programación Dinámica o Backtracking** que maximice el beneficio de los trabajos que se procesen.
- 115.—Joseba Cilón y Tom Tolaba son dos espías profesionales, y han recibido el encargo de una peligrosa misión en Irak. Se trata de desvalijar un almacén dónde se guardan N cajas llenas de pequeños recipientes que contienen el líquido usado por el ejército de Sadam para fabricar el gas mostaza. Cada una de las cajas tiene un determinado peso, P_i . El ejército iraquí ha descubierto los planes de los espías y se dirigen al almacén. Joseba y Tom han recibido la alarma de sus contactos y saben que deben huir rápidamente. Aunque llevan mucho tiempo trabajando juntos, no se llevan nada bien, y ninguno de ellos está dispuesto a trabajar más que el otro para luego cobrar la misma recompensa, por lo que se deben repartir la carga en dos partes exactamente iguales, pero como no tienen tiempo para abrir las cajas, deciden utilizar el peso de las mismas para repartirse la carga. En caso de que el reparto no pueda hacerse en dos partes iguales, cada uno prefiere perder la recompensa antes que trabajar más que su “amigo”, por lo que si se da el caso, huirán de vacío. Diseñe un algoritmo basado en **Programación Dinámica o Backtracking** que resuelva el problema.
- 116.—**La Herencia.** El abuelo *Milbi Lletes* está en las últimas y sus dos nietos le están insistiendo constantemente para que le cuente qué parte de su inmensa fortuna les deja. El abuelo por su parte no suelta prenda, pero sí les comenta que les dejará una parte de su fortuna a partes iguales.
- Días más tarde el abuelo pasa a mejor vida y el notario hace presencia en la mansión "*Villa Billete*". Una vez leído el testamento y llegando a la parte en la que a los nietos se hace referencia, el notario lee: “Yo *Milbi Lletes* quiero dejar todo lo contenido en esta lista de bienes indivisibles, cada uno tasado según la relación que se adjunta, a mis dos nietos, *Loqui Erotodo* y *Paca To*. El reparto, que queda en manos del notario, debe ser lo más equitativo posible.”.
- El notario viendo lo que tardaría en cuadrar dicho problema, inteligente donde los haya, decidió llamar a dos estudiantes de Ampliación de la Programación para resolverlo. Éstos bien aleccionados en sus clases, le preguntaron que en caso de no ser exacta la división qué debían hacer. El notario tras pensarlo decidió que el mayor de los dos tuviera la mayor parte.

Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que realice el reparto de las pertenencias.

- 117.–Abundio, después de su "idea feliz" de vender el coche para comprar gasolina, se ha dado cuenta que lo que hizo no estuvo del todo bien y quiere darle salida a toda esa gasolina que ahora no le sirve para nada. Para ello irá por las N casas de sus amigos (Pichote, Tom Tolaba, el que pellizcaba ventanas,...), con el fin de venderla toda y conseguir el dinero que se gastó. Cada amigo le comprará una cantidad c_i distinta de gasolina, según la necesidad de éste. Abundio conocedor de su pueblo, sabe la distancia en tiempo entre las casas de sus amigos por lo que no le será difícil elegir una ruta a seguir.

Pero existe un problema: Abundio, que no se fía de su instinto, quiere vender toda la gasolina antes que empiece su película favorita en televisión, "Dos tontos muy tontos", ya que ésta le puede dar alguna idea de qué hacer con ella. Por ello, suponemos dispondrá de un tiempo T para vender la gasolina.

Ayude a Abundio diseñando un algoritmo de **Programación Dinámica** o **Backtracking** que devuelva la mejor ruta a seguir con el fin de vender la máxima gasolina posible en el tiempo del que se dispone. **NOTA:** Disponemos de un depósito con ruedas en el que cabe todo el combustible y que vamos desplazando de una casa a otra.

- 118.–El Jaro "Mantalombro" es el alumno más "gorrón" de la Universidad. Su dedicación durante todo el día es estar tumbado en el sofá jugando a las consolas. Recientemente ha adquirido una consola "Polystation" de última generación, ya que había conseguido pasarse todos los juegos de las otras consolas que poseía. Hasta aquí todo bien, pero no había caído en un pequeño detalle y es que no poseía ningún juego para esta nueva consola y no está dispuesto a perder ni un solo minuto, ya que su cerebro le solicita jugar en todo momento. Su idea es la siguiente: sus "contactos" han localizado a N posibles "pardillos" de los cuales puede aprovecharse para grabar juegos compatibles con la "polystation". Cada pardillo p de estos posee m_p juegos y los contactos estiman un tiempo t_i en grabarlos. Plantee una estrategia basada en **Programación Dinámica** o **Backtracking** que le permita al "Jaro" grabarse la mayor cantidad de juegos distintos suponiendo que dispone de un tiempo máximo T .

- 119.–Ante la inminente llegada de la Semana Santa, el Ayuntamiento –quizá conmovido tras años de insistencia, o quizá ávido de votos ante la proximidad de las elecciones– ha decidido por fin, permitir la salida de nuestro paso: el de la *Virgen de los Amargados por Ampliación*.

Por ser nuestro primer año, el alcalde nos concede el beneficio de elegir la ruta por la que irá nuestra procesión. La única restricción que nos impone es el tiempo máximo T que nuestro paso puede estar en la calle.

Por suerte, contamos en nuestra cofradía con Bernabé Ato Acérrimo, vieja gloria de los matriculados en Ampliación, y gran conocedor de la cultura religiosa. Bernabé ha puesto a nuestra disposición un mapa del pueblo, donde se detalla la distancia en tiempo entre los balcones de los saeteros que tiene el pueblo. También sabemos el nivel de ruido r_i de cada saetero, y el tiempo que estará cantando t_i cada saetero. Sabiendo que queremos empezar y

acabar la procesión en la Iglesia del pueblo (las distancias en tiempo desde cada balcón a la Iglesia, así como las de cualquier balcón hasta la Iglesia, son conocidas), supuesto que no pasaremos dos veces por el mismo balcón, diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que nos dé la ruta que debemos seguir para que nuestra procesión sea la que más ruido “meta” esta Semana Santa.

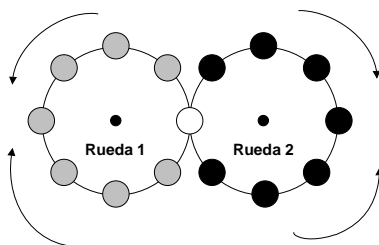
120.–Dado un plano del metro, supuesto que conocemos el tiempo que tarda cada tren en recorrer cada trayecto entre dos estaciones contiguas de una misma línea, el tiempo que está parado en el andén en cada estación, y también, el tiempo que tardamos en efectuar un transbordo de una línea a otra en cada una de las estaciones en las que esto es posible. Diseñe algoritmos que resuelvan los problemas que se indican a continuación:

- Determinar el camino mínimo (en tiempo) entre dos estaciones dadas de la red. Utilice **Programación Dinámica**.
- Resuelva el problema del apartado a), supuesto que no podamos hacer más de k transbordos. Utilice **Backtracking**.

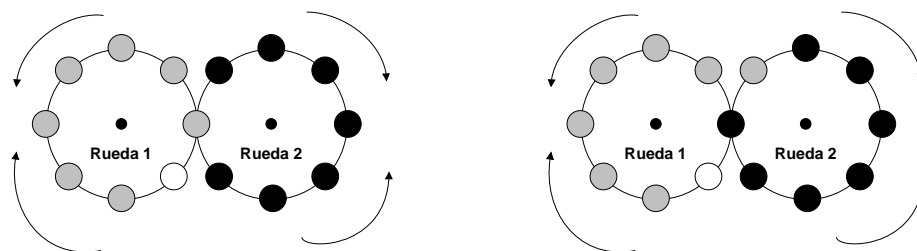
121.–**Cruzar un Puente Estrecho y Peligroso.** Tenemos un grupo de N personas que tienen que cruzar un puente por el que solo pueden ir en fila de a uno, y que además, no soporta más que el peso de dos personas. Sabemos que el grupo solo dispone de una lámpara para alumbrarse y que cada persona p tarda un tiempo t_p en cruzar el puente. Además, cuando dos personas cruzan el puente tienen que llevar la lámpara consigo, y el tiempo que emplean en cruzarlo es el mayor de las dos personas.

Diseñe un algoritmo que devuelva cómo deben cruzar el puente para que el tiempo que empleen en ello sea mínimo. Aplíquelo al caso de un grupo de cinco personas, cuyos tiempos en cruzar el puente de cada componente del grupo son: 1, 3, 6, 8 y 12 minutos.

122.–**Puzzle de las ruedas.** Diseñe un algoritmo que sea capaz de resolver el puzzle de las ruedas. Este puzzle se compone de dos ruedas giratorias y N bolas por cada rueda. De tal manera que al final del algoritmo queden en la siguiente configuración:



Para ver el funcionamiento, observe que tras una giro de la rueda 1 en sentido horario y otro de la rueda 2 en el mismo sentido las configuraciones serían:



123.–Las cartas enviadas por los N niños de un albergue se han extraviado, y no han llegado ni a *Santa*, ni a *Los Reyes Magos*. Aunque éstos por otras colaboraciones saben de las preferencias de juguetes de los niños. Así, tienen datos p_{ij} que les indican la preferencia que tiene el niño i por el juguete j . Tienen el problema que sólo disponen de J juguetes para los N niños, aunque éstos han pedido en sus cartas bastantes juguetes más, pero afortunadamente $J \geq N$. Diseñe algoritmos basados en **Programación Dinámica** o **Backtracking** que resuelvan cada uno de los problemas que siguen:

- a) Supuesto que reparten un juguete por niño:
 - i) Devuelva la asignación que maximice la suma de las preferencias.
 - ii) Devuelva la asignación que hace posible tener entretenidos a todos los niños el máximo de tiempo posible, puesto que un niño juega con un juguete tanto tiempo como indica su preferencia, tiempo que está entretenido con el juguete, y transcurrido ese tiempo se “cansa” del mismo y pasa a aburrirse o a incordiar al resto de niños. Todos los niños reciben los juguetes al mismo tiempo.
- b) Resuelva ii) supuesto que $N \leq J < 2N$ y se pueden repartir hasta DOS juguetes por niño.

124.–*Homer* ya está harto de que su malévolo jefe, el señor *Burns*, le “mangonee” continuamente, y ha jurado por el Dios *Duff* que esta vez va a ser la última. La última jugada que le ha hecho el señor *Burns* a *Homer* es robarle el juguete preferido por su hija *Maggie*, el oso Bobo. Bien es cierto que este oso perteneció en un tiempo muy lejano al señor *Burns*, pero cada vez que lo vuelve a encontrar lo vuelve a perder. *Homer* ha decidido hacerle probar un poco de su propia medicina al señor *Burns*, y siguiendo los “buenos” consejos de sus camaradas *Carl* y *Lenny*, *Homer* está dispuesto a asaltar la mansión del señor *Burns* y robarle la mayor cantidad posible de joyas que éste guarda en N baúles dentro de la habitación 17 del tercer piso. El problema está en que *Homer* sólo dispone de un tiempo T antes de que el fiel sirviente del señor *Burns*, *Smithers*, vuelva para arrojar a su querido jefe. Pero este no es el único problema, ya que *Homer* está obsesionado últimamente con su suerte y piensa que asaltar un baúl es tarea fácil, pero que hace que la suerte de *Homer* se tambalee. Además, *Homer* en un esfuerzo inigualable, ha conseguido pensar que si pide consejo a alguien más sabio que él, es más probable que obtenga un mayor número de joyas. Para ello *Homer* ha recurrido a su querida hija *Lisa*. Ayude a *Lisa* a diseñar un algoritmo basado en **Backtracking** o **Programación Dinámica**, de forma que *Homer* consiga robar el mayor número posible de joyas dentro del tiempo que dispone y sin “tentar” a la suerte de la que dispone siendo conocido las joyas que hay en cada baúl, el tiempo que se tarda en asaltar cada uno de ellos y la suerte que menguará a nuestro amigo *Homer*.

- 125.–**El bar de Moe.** En la ciudad de *Springfield* es costumbre para muchos trabajadores de la central nuclear (entre ellos *Homer Simpson*) pasar la noche de los viernes en el bar de *Moe* tomando cañas sin parar hasta acabar tirados por los suelos. Pero *Moe* últimamente ha tenido muchas peleas dentro del bar y ha perdido muchas jarras en las que sirve cerveza. Como todos saben *Moe* es muy rácano, y por eso no quiere comprar más jarras. Por lo que de los N clientes que pueda tener en una noche solo podrá atender a M de ellos al mismo tiempo ($M < N$) siendo M es el número de jarras de que dispone. Por supuesto cada cliente deja una propina p_i diferente a parte del precio de cada jarra y tarda un tiempo t_i determinado en beberse la cerveza.

Ayude a *Moe* a maximizar su beneficio, diseñando un algoritmo basado en **Programación Dinámica** o **Backtracking** para determinar a qué clientes tiene que servir cerveza, siempre y cuando no se sobrepase un tiempo T , ya que el jefe de policía *Clancy Wiggum* le obliga a cerrar a una cierta hora o sino le cierra el local.

- 126.–**Pluma Blanca** es un pequeño indio que ya tiene edad para pasar a una edad adulta. Sólo le falta la prueba que probará ante los demás miembros de su tribu su madurez. Se trata de dejar a *Pluma Blanca* sólo en el desierto con una pequeña cantimplora de agua y esperar que vuelva al poblado sano y salvo trayendo consigo, un mechón de pelo del puma *Malo Malo*, una ramita del viejo árbol del arroyo en el que se mira la *Luna*, un pedazo de roca del *Monte Olvidado*, una flor de más allá del lago y una serpiente de las *Tierras Lejanas*. Todos estos destinos están muy lejos unos de otros y por lo tanto en algún punto *Pluma Blanca* deberá volver a rellenar su cantimplora. Durante largo tiempo los ancianos de su poblado le han hablado de multitud de fuentes naturales a lo largo de los diversos lugares que tiene que visitar, y *Pluma Blanca* los ha aprendido al igual que las distancias que hay entre estos y los lugares a los que tiene que ir.

Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que le permita a *Pluma Blanca* encontrar todos los objetos para demostrar su madurez, sin deshidratarse y sin tardar más de un tiempo T del que dispone antes de la próxima luna llena.

- 127.–**El Pirata Barbarroja.** El pirata *Barbarroja*, famoso en el mundo entero por lo malo que llega a ser con sus enemigos, ha encontrado a un náufrago en una isla y en lugar de acogerlo en su barco lo ha dejado en mitad del Mar Caribe, dentro de un pequeño barril y con un mapa. El mapa le indica los posibles recorridos que puede hacer para llegar al país de "*Nunca Jamás*". Pero *Barbarroja* le ha advertido que durante ese recorrido puede caer en las manos de otros piratas más malvados que él, como por ejemplo el *Capitán Garfio*, y lo que sería peor, en las fauces de grandes tiburones sedientos de sangre.

El náufrago tiene siete vidas, por lo que podría salir de siete piratas malvados o tiburones pero lo está pasando muy mal porque se aburre mucho en mitad del mar y está deseando salir lo antes posible de allí. Ayude al náufrago diseñando un algoritmo basado en **Programación Dinámica** o **Backtracking** para que tarde lo menos posible en llegar a "*Nunca Jamás*" sin que pierda sus siete vidas, utilizando el mapa que le ha dado el pirata *Barbarroja*.

- 128.–**El Ladrón Astuto.** El “*Pies Largos*” está pensando en hacerse ladrón profesional, sabe algo de programación y ha pensado obtener un algoritmo que le ayude en su tarea.

Ha realizado una lista de N posibles joyerías en las que robar y ha estudiado detenidamente el plan para robar en cada una de ellas. Dispone de un plano en el que se especifica el tiempo que tarda en desplazarse entre las joyerías, y para cada joyería tiene el tiempo que tardará en robarla, y el botín que obtendrá. Cada joyería tiene un sistema de seguridad (medio, alto, excelente), y “*Pies Largos*” las ha clasificado atendiendo al mismo. Además, estima que la policía tardaría un tiempo T en localizarle y detenerle, esté donde esté, tiempo que empieza a contar desde el momento en el que comience sus fechorías.

Diseñe un algoritmo basado en **Backtracking** o **Programación Dinámica** para maximizar el botín (volver a su casa antes de tiempo sin ser detenido).

- 129.–En vista de los malos resultados que están obteniendo los alumnos de Ampliación de Programación, los dioses están enfadados y quieren dar un escarmiento divino a los profesores de dicha asignatura.

Por ello, *Hades*, dios del infierno, *Ares*, dios de la guerra, y *Apolo*, dios de la razón y mensajero de los dioses se han reunido para buscar la mejor forma de “persuadir” a dichos profesores para que cambien su proceder, antes de ser sometidos a un castigo divino de mayores dimensiones.

Cada uno de los dioses tiene una idea de cómo dar un “pequeño” escarmiento a los descarriados profesores: *Hades* promulga las ventajas de pasar unas vacaciones en el averno al cuidado de su precioso perrito cerbero. *Ares*, propone crear un conflicto bélico en la Escuela de Informática y hacer que los Cuatro Jinetes del Apocalipsis campeen por los despachos de todos los profesores. *Apolo*, el más razonable y con una increíble simpatía por los mortales, propone encontrar medidas menos radicales. Tras una larga deliberación, *Apolo* ha convencido a sus compañeros de que la mejor forma de resolver este conflicto es enviando a *Tyche*, diosa de la Fortuna y el Destino, a visitar a los profesores, de manera que les proporcione algo de “mala” suerte. Sin embargo, *Apolo* no había contado con los caprichos de *Zeus*, pues éste no dejará salir del *Olimpo* a *Tyche* durante una noche más que un tiempo T , para que realice su labor.

Diseñe un algoritmo de **Programación Dinámica** o **Backtracking** que ayude a *Tyche* a visitar al mayor número de los N profesores de la asignatura de Ampliación de Programación. Sabiendo que solo dispone de un tiempo T y que cada profesor p necesita un tiempo de visita distinto t_p . Además, se ha elegido una noche en la que cada profesor pasará toda la noche en su casa, o en su despacho, y en la que no coinciden dos profesores en un mismo sitio. Para ayudarse en su tarea, *Tyche* dispone de un mapa en el que tiene las distancias en tiempo para ir de uno a otro de los N lugares posibles que puede visitar que es en los que se encuentran los profesores en esa noche oscura... para algunos.

- 130.–**La Expo de Zaragoza.** Un grupo de amigos, se ha propuesto visitar el mayor número pabellones de la Expo del Agua. Para cada uno de los N pabellones se sabe el tiempo que tendrían que esperar en la cola antes de entrar y el tiempo que tardarían en visitarlo. Se sabe

además, la distancia en tiempo que se tarda en desplazarse entre dos pabellones cualesquiera y también la distancia (en tiempo) desde la entrada de la Expo a cada pabellón. Desean visitar **el mayor número de pabellones en un día**, supuesto que disponen de H horas. Diseñe un algoritmo basado en *Programación Dinámica* o *Backtracking* que resuelva el problema.

- 131.—Flipi Flipao quiere comprarse un coche deportivo, de esos que flipas, pero no tiene suficiente con el dinero que gana en su curro. Por eso se ha buscado otro para obtener un plus y conseguir su "sueño". Con el pluriempleo se ven resultados.

Va a repartir los folletos del partido *LEP* (*Ladrones en Paro*) que se presentará en las próximas elecciones, pero claro su trabajo habitual le ocupa casi todo el tiempo, y para repartir dichos folletos solo dispone de un tiempo T . Cuantos más folletos reparte, más dinero le pagan.

Flipi se ha hecho de un plano que le dice la distancia entre dos urbanizaciones, y también la distancia entre cada urbanización y la central en la que debe recoger los folletos. Además, ha realizado sus averiguaciones y sabe cuántos folletos repartirá en cada urbanización, y el tiempo que tardará en repartirlos.

El problema está en los partidarios radicales del partido *TU* (*Timadores Unidos*), rivales del *LEP*, de éstos se sabe que organizan “partidas” para “informar” convenientemente a cualquier repartidor de propaganda ajena que no debe hacerlo. Al último repartidor que se topó con ellos, todavía le duelen los huesos...

Pero Flipi Flipao no es un gil, y sabe de antemano el recorrido de la partida, concretamente, los instantes inicial y final, que la “partida” estará en cada urbanización (puede ser que no vayan a todas las urbanizaciones).

Diseñe un algoritmo que ayude a Flipi a **repartir el mayor número de folletos en el tiempo T de que dispone**.

- 132.—En una ciudad el alcalde ha decidido asfaltar sus calles por motivos electorales, pero tiene el problema de que no puede asfaltar todas sus calles por motivos de presupuesto. Ha decidido asfaltar aquellas calles que unan una plaza con otra, quedando todas las calles asfaltadas unidas. Para cada calle y cada plaza sabe cuánto cuesta asfaltarla, y también su tiempo en hacerlo. Además, sabe por estadísticas de elecciones anteriores y encuestas realizadas recientemente, el número de potenciales votantes de su partido tanto para cada calle, como para cada plaza.

El alcalde se ha propuesto maximizar el número de sus partidarios con plaza o calle asfaltadas, puesto que cree que contribuirá a conseguir afianzar el voto entre los seguidores acérrimos, y en el resto, a inclinar definitivamente la balanza a su favor. Diseñe un algoritmo basado en *Programación Dinámica* o *Backtracking* que indique cuáles son las calles y plazas a asfaltar, supuesto que disponemos de un presupuesto P .

- 133.—Tenemos un plano de una ciudad en el que tenemos señaladas las posibles paradas de autobús y las distancias entre ellas. El conjunto P de las posibles paradas se ha elegido de tal forma que dado un punto cualquiera de la ciudad la distancia a dicho conjunto no sea superior a d .

Se trata de establecer la **ruta circular de menor longitud** que debe seguir un autobús para que cualquier parada de la misma quede a una distancia no superior a D de la parada más cercana del autobús. Con ello tendríamos que cualquier punto de la ciudad a un punto de la ruta no estaría a distancia superior a $D+d$. Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** para resolver el problema.

- 134.— “**Yo no soy Tonto**” (*¡Quien lo diría!...*) Una conocida cadena de electrodomésticos, famosa por sus agresivas campañas publicitarias, ha tenido que devolver el 25% del precio de todos los televisores vendidos entre el 1 de mayo y el 20 de junio, debido a su compromiso publicitario a devolver el “cuarto” del precio del televisor si la Selección Española de fútbol pasaba de “cuartos” en la Eurocopa 2008. Afortunadamente para España y desafortunadamente para los del departamento de publicidad, España pasó.

Firmemente convencidos de su estrategia, los “fantásticos” publicistas convencieron al equipo directivo de que redoblara su apuesta e idearon otra campaña “**1x2**” en la que si los clientes compraban, y hasta un día antes de la final, cierto paquete de electrodomésticos, y España ganaba la final, se les regalaba otro paquete de electrodomésticos del mismo importe que servirían a domicilio sin coste adicional para el cliente. Y el caso es que la Selección, tras años de “pertinaz” sequía, ha ganado la Eurocopa.

Se ven ahora con que tienen que servir N paquetes a otras tantas casas (un paquete por casa) y hacerlo cuanto antes, puesto que la clientela se impacienta ante los rumores de que están intentando “echarse para atrás”, al hacerse pública una reprimenda del Director General a los publicistas en la que les decía que “*no eran tontos, eran ...*”. Parece que alguien grabó con un móvil un vídeo con la “escena” y lo ha puesto a circular por Internet, aunque nadie ha podido confirmar su veracidad.

Se sabe la distancia entre todas las casas, y también entre el almacén central y cada casa. Lamentablemente por la imprevisión, sólo disponen una pequeña furgoneta en la que caben hasta M objetos.

Se trata de repartir los N paquetes a las casas en el menor tiempo posible. Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que resuelva el problema. **NOTA:** Puede suponerse que sólo se regresa al almacén cuando la furgoneta está vacía, y que cuando regresa al almacén, se carga al máximo.

- 135.— “*Estoy Queme Rompo*” se ha metido en el mundo de la publicidad. Ha montado una empresa de marketing “*Elmas Guapo S.A*” y está buscando una serie de actores que plasmen su último proyecto de nueva creación durante la presentación del producto a los medios de comunicación.

Se han presentado N actores a las pruebas, y según los resultados de las mismas se conocen de antemano la efectividad e_i de un actor al realizar una de las M posibles presentaciones, y el tiempo t_i que emplea en realizarla. Además, ha comprobado que el efecto de una exposición

se ve reforzado o disminuido en un porcentaje según la exposición realizada inmediatamente antes, y que es independiente del actor que la realiza. La tabla r_{ij} indica el refuerzo o disminución que experimenta la exposición j realizada inmediatamente después de la exposición i . Considérese la tabla r_{ij} como una constante multiplicativa de la efectividad (mayor que uno positiva, menor que uno negativa).

Se trata de determinar la composición de la presentación, para que **el acto resulte lo más efectivo (convinciente) posible**, supuesto que cada actor no realizará más de una exposición y que el acto social deberá durar como máximo T unidades horarias. Diseñe un algoritmo que resuelva el problema.

- 136.—**El Jardín Oriental.** Tenemos un jardín oriental que pretendemos decorar de tal manera que consigamos un equilibrio del *Chi* según las enseñanzas del *Ikebana*. Según el maestro *Tzin-Tze Tou*, nuestro jardín debería contar con una combinación de colores amplia, pero esto no siempre es posible por ciertas limitaciones, principalmente monetarias.

En nuestro jardín tenemos N jardineras y cada jardinera está dividida en M zonas. No todas las zonas son iguales, E_{ji} es el espacio de la zona i de la jardinera j . Podemos elegir entre P plantas distintas. Cada planta p tiene color c_p y un precio p_p , y necesita de un espacio e_p (en cm^2) de la jardinera. En cada zona solo se puede plantar una planta. Nuestro objetivo es decorar todas las jardineras del jardín, de manera que no haya dos plantas del mismo color en una jardinera y procurando gastar lo menos posible.

Diseñe un algoritmo basado en **Backtracking** que nos indique cuál es la colocación de plantas óptima (menor gasto) para conseguir nuestro objetivo.

- 137.—Dado un tablero cuadrado de N filas, en cada una de sus casillas hay un indicador luminoso, que está apagado (**OFF**) o encendido (**ON**). Al pulsar en una casilla, esa casilla y todas sus adyacentes cambian su estado (de apagado a encendido y viceversa).

Diseñe un algoritmo basado en **Backtracking** que determine si es posible apagar todo el tablero, sin pulsar dos veces una casilla. Nótese que alguna casilla puede no pulsarse ninguna vez.

Si es posible apagar el tablero, **determine la secuencia mínima de pulsaciones que apagan el tablero.**

- 138.—Un grupo de amigos, formado por n parejas (n mujeres y n hombres), se reúnen para cenar. El protocolo del buen comensal exige que, a la hora de sentarse a la mesa (redonda), hombres y mujeres deben alternarse y que nadie debe sentarse al lado de su pareja habitual. Para que la velada sea lo más agradable posible, hay que **maximizar el grado de bienestar total**, obtenido sumando los grados de afinidad mutuos entre los comensales sentados en posiciones adyacentes. Al efecto, se dispone de sendas matrices que nos indican la afinidad entre hombres y mujeres y entre mujeres y hombres. Una vez establecida cada pareja de vecinos, la afinidad mutua se calcula multiplicando la de cada miembro por el contrario. Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que encuentre una solución óptima al problema.

- 139.—El príncipe de un lejano país ha encontrado a su media naranja y al fin ha decidido casarse. Lo ha tomado como un negocio para sacar la mayor pasta posible, y además de poder pagarse el viaje de bodas, que le sobre para pagar el pequeño palacete que se ha construido.

Ha decidido invitar al mayor número de altos dignatarios y demás gente de postín con la idea de que sus regalos estarán a la altura de lo que de ellos se espera. Ahora bien, éstos no están acostumbrados a regalar dinero, y menos aún a desplazarse sin su propio séquito, o no contar con unas mínimas medidas de seguridad. Así cada uno ha demandado tener un número mínimo de agentes que cuiden de su seguridad y hospedarse en un sitio acorde a su *status*. Aunque hay algunos que solo tendrán necesidades de seguridad ya que su viaje será breve estando acompañando a los novios en la ceremonia y el banquete.

El príncipe ha hecho cuentas y ha visto que no tiene suficientes “maromos” para cubrir las necesidades de los invitados, y tampoco es que disponga de demasiadas “residencias”.

Determinado como está a optimizar recursos para conseguir sacar la mayor tajada del magno acontecimiento, ha reunido a sus asesores para que le proporcionen la mejor asignación de recursos. Los asesores acostumbrados a “dorarle la píldora” no están por la labor de tener que realizar esfuerzos mentales de tal clase. Así, por unanimidad han resuelto recomendar que sea el recientemente creado Departamento Informático del ministerio de Recaudación, Información y Planificación (RIP), que a algunos de los asesores les está ocasionando algún que otro dolor de cabeza, que **diseñe un programa de devuelva la asignación óptima de recursos para conseguir que asistan los invitados que proporcionen los mayores ingresos**. NOTA: El citado departamento está instalado en las mazmorras del castillo y en él trabaja algún que otro alumno de Ampliación, ya familiarizado con las condiciones de trabajo.

- 140.—Con la finalización de las obras de los nuevos edificios, la dirección de la ESI tiene que llevar a cabo una reasignación de los despachos a sus profesores. Tenemos N profesores y M despachos. Los despachos son individuales o permiten dos puestos. Naturalmente, un despacho de dos puestos puede también ser asignado a un solo profesor, pero bajo ningún concepto un despacho individual se asignará a dos profesores.

Cada profesor k tiene una categoría laboral $CatProf_k$ y cada despacho es apropiado para hasta una categoría $CatDesp_k$ donde la categoría 0 representa la inferior y la $CatMax$ la mayor. A un profesor de una categoría nunca se le asignará un despacho de categoría inferior.

Entre las restricciones se tiene que cada profesor ha enviado a la dirección una lista de profesores con los que le resulta imposible compartir despacho. Además, la dirección sabe que algunos profesores deben ocupar despacho de forma individual.

Diseñe un algoritmo que determine la asignación que minimiza el número de despachos utilizados.

- 141.—En la empresa Losa Preta, O.S. andan escasos de espacio y necesitan distribuir los trabajadores de una forma óptima. Disponen de M puestos de trabajo y N trabajadores. Cada trabajador i tiene unas restricciones horarias r_i , una carga de trabajo c_i y una preferencia p_{ip} por cada puesto p de trabajo. Naturalmente en un puesto no pueden estar trabajando al mismo tiempo dos trabajadores distintos.

Diseñe algoritmos basados en **Programación Dinámica** o **Backtracking** para determinar el horario de tal forma que todos los trabajadores cubran su carga de trabajo y se **maximice la preferencia del conjunto**, según las dos alternativas siguientes:

- a) Un trabajador sólo trabaja en un puesto. Esto es, si se le asigna el puesto k , entonces solo trabaja en el puesto k .
- b) Un trabajador puede trabajar en varios puestos en distintas sesiones, pero no cambia en sesiones consecutivas. Por ejemplo, si de 9:00 a 10:00 trabaja en el puesto k , y si trabaja también de 10:00 a 11:00, entonces también lo hará en el puesto k .

Naturalmente puede ocurrir que las restricciones sean demasiado fuertes y el problema no tenga solución en los términos planteados. En tal caso, determine **cuál es el mínimo número de puestos de trabajo necesarios para satisfacer las restricciones horarias**, y supuesto que los empleados tienen una preferencia nula por los nuevos puestos añadidos determine la **distribución que maximice la preferencial global**.

NOTAS: La carga de trabajo es el número de horas semanales que tiene que trabajar. Las restricciones horarias se entiende que son las horas a las que puede trabajar durante la semana. La preferencia dada se entiende por unidad de tiempo.

142.—A *Frodo Bolsón* le ha tocado la pesada carga de llevar el antiguo anillo de su tío *Bilbo* desde la lejana tierra de la comarca hasta la posada del *Pony Pisador*. Su gran amigo el mago *Gandalf* le ha advertido de los grandes peligros que le acechan por el camino y de que puede utilizar el poder del anillo solamente durante cinco ocasiones antes de ser descubierto por el *Señor Oscuro*. Así, ha decidido idear un planteamiento basado en **Programación Dinámica** desarrollado durante largo tiempo por los grandes señores *elfos* y transmitido hasta él para hacer más seguro su viaje. Para ello, *Frodo* se ha provisto de un mapa de la comarca con los diferentes lugares por los que puede pasar, y las distancias en tiempo entre ellos, en el que ha anotado los posibles lugares donde estarán los malvados *orcos* esperándole. Diseñe un algoritmo basado en **Programación Dinámica** para encontrar el camino mínimo hasta el *Pony Pisador*.

143.—En la *Cima de los Vientos*, la compañía del anillo ha sufrido el ataque de los temibles *Nazgûl* como consecuencia a la imprudencia de los *hobbits*. El resultado ha sido que el portador del anillo, *Frodo*, se debate ahora entre la vida y la muerte por el cuchillo de *Morgul* que tiene clavado cerca del corazón.

Aragorn sabe que en ese estado no durará mucho y antes de partir hacia *Rivendel*, debe preparar un ungüento que frene un poco el avance de *Frodo* hacia el camino sin retorno, hacia la muerte.

Para preparar dicho ungüento debe buscar M hierbas curativas, entre los N lugares más cercanos que conoce, sabiendo que en cada lugar crece un único tipo de planta de las M necesarias ($M < N$). Ahora lo importante es conseguir un remedio que consiga hacer que *Frodo* gane algo de fuerzas hasta llegar al refugio. *Aragorn* como buen montaraz, conoce la distancia, en tiempo, a cada uno de los N lugares, y el tiempo que puede llegar a tardar en cada uno, para buscar la hierba curativa que crece en dicho lugar. Debemos ayudar a *Aragorn*

con el uso de **Backtracking** o **Programación Dinámica** para que consiga las M hierbas, en el menor tiempo, de forma que pueda volver a la *Cima de los Vientos* lo antes posible, pues no sabe cuanto tardarán en volver los *Nazgûl*, pero volverán.

- 144.—Nuestros amigos, tras el triunfo del *Ejército del Oeste* sobre las tropas del *Señor Oscuro*, *Sauron*, en la guerra por *La Tierra Media*, deciden regresar a la ciudad de *Minas Tirith* para celebrar la victoria. Al anochecer celebrarán en el palacio una fiesta en honor del portador del anillo, *Frodo Bolson*. Cansados por los preparativos, *Merry* y *Pippin*, dos *hobbits* de reputación impecable y expertos catadores de cerveza, deciden ir a “explorar la ciudad” visitando las tabernas de la ciudad, acompañados por enano *Gimli*.

Faramin, *Senescal de Gondor*, les ha proporcionado un mapa turístico de la ciudad, con las N tabernas de la Ciudad en el que se indica, además del tiempo que se tarda en llegar a una taberna desde otra cualquiera, el precio de la jarra de cerveza en cada taberna, y el “ambiente” que se respira en la taberna. Como los *hobbits* son muy fiesteros, dependiendo del “ambiente” de la taberna, tardan más o menos tiempo en beberse la cerveza. *Gimli*, sin embargo, siempre tarda lo mismo en beberse una cerveza sea de la taberna que sea. Los tres no abandonan la taberna hasta que todos han terminado con su cerveza. Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que resuelvan los problemas que siguen:

- a) Suponiendo que disponen de T horas hasta el anochecer y que tienen D dinero cada uno, se trata de maximizar el número de tabernas visitadas, suponiendo que en cada taberna se toman una cerveza.
- b) Resuelva el apartado anterior teniendo presente las siguientes consideraciones:
 - i) cada taberna tiene una medida propia de jarra de cerveza,
 - ii) cada uno de los “exploradores” tiene un “aguante” determinado, i.e., es capaz de beber una cantidad de cerveza dada, sin emborracharse, y que
 - iii) en el momento en el que uno se emborracha deciden volver al palacio.

- 145.—Es de noche y una horrenda criatura vaga por las zonas pantanosas de la *Tierra Media* cobijada por la oscuridad. Este ser una vez fue un *hobbit*, que respondía al nombre de *Smeagol*, pero ahora se le conoce como *Gollum* por el inquietante sonido que produce al engullir pescado crudo, su único alimento. Las gentes del lugar que aún recuerdan cómo era *Gollum* cuando todavía era *Smeagol* coinciden en que era de carácter afable e incluso estaba estudiando Ampliación de la Programación en *Hobbiton* pero una extraña influencia maligna terminó por ennegrecer su pequeño corazón.

“El *hobbit* malo nos ha robado nuestro tesoro, ahora el buen *Smeagol* no podrá ir a pescar porque sin tesoro, los sucios orcos le darán caza. Ahora sólo podré viajar entre las charcas para cazar pescado mientras dure la noche y los orcos duerman” – repetía una y otra vez *Gollum*.

Son muchos años los que lleva *Gollum* por la zona y ha terminado por aprenderse la distancia entre las charcas donde está el pescado y la cantidad de peces que puede encontrar en cada una, además del tiempo que tarda en pescar los peces de esa charca. *Gollum* debe cargar con todos los peces que ha pescado, lo que le supone que si en un trayecto entre dos charcas

tardaba de vacío t , al desplazarse entre ellas cargado con p peces, tardará $t(1+kp)$, donde k es constante expresada en tantos por uno.

Diseñe un algoritmo basado en **Backtracking** o **Programación Dinámica** que ayude a *Gollum* a obtener la mayor cantidad de peces posible sabiendo que tiene que estar de vuelta en su cueva antes de que amanezca para lo cual falta un tiempo T .

146.–**Lectura de don Quijote.** Tenemos N lectores que van a leer otros tantos fragmentos del Quijote. El lector l lee el fragmento f con una intensidad i_{lf} , y tarda en leerlo un tiempo t_{lf} . **Dado un tiempo máximo T se trata de conseguir la máxima intensidad de lectura.** Resuelva el problema para cada uno de los siguientes apartados:

- a) Cada lector leerá un fragmento.
- b) Si tenemos M lectores, ($M < N$), devuelva la asignación que determina la máxima intensidad, respetando que si a y b son dos lectores cualesquiera, y n_a y n_b el número de fragmentos leídos por cada uno de ellos, se verifique que $|n_a - n_b| \leq 1$.

147.–Don Quijote y Sancho han vuelto al lugar del que partieron. Y aunque el cura, el ama y el bachiller quemaron los libros de los aposentos de don Quijote. Éste ha dado con unos baúles que dejó olvidados un viajero de aspecto triste, y ha encontrado una colección de *Novelas de Caballerías*, alguna de las cuales hace mención a las hazañas de cierto caballero andante por las tierras de La Mancha.

Alonso Quijano, ha colocado los libros encontrados en una estantería, y ha comenzado a leerlos según el orden en el que se encuentran en la estantería. Si bien, ansioso como estaba por saber de las aventuras de caballeros andantes, no ha reparado en el orden de colocación de los libros en la estantería.

El cura y el bachiller, se temen lo peor. Saben que no pueden volver a quemar los libros, porque Alonso Quijano se volvería loco si de un día para otro desaparecieran los libros, pero han advertido, como se ha dicho más arriba, que Alonso lee los libros según el orden de colocación en la estantería. Y creen haber encontrado una estrategia para demorar en lo posible que se vuelva loco: **reordenarán los libros que aún le quedan por leer a Alonso Quijano para demorar al máximo que se vuelva don Quijote.** A continuación, el bachiller Sansón Carrasco partirá a Salamanca en busca de sabios doctores en busca de consejo para la enfermedad que aqueja a nuestro hombre, con el margen que les proporciona la reordenación de los libros.

Tras averiguaciones varias, han dado con otros casos documentados del mismo mal que aqueja a Alonso Quijano, y han logrado saber para cada libro i el tiempo t_i (en días) que tardará en leerlo, y el desorden psicológico (locura) l_i que le causará su lectura (sólo se la causa cuando ha completado su lectura).

Sabiendo que Alonso Quijano está a L pasos de convertirse en don Quijote, diseñe un algoritmo basado **Programación Dinámica** o **Backtracking**, que determine la **ordenación de los libros que proporcione al bachiller el mayor tiempo** para completar la misión de

encontrar remedio al mal que acecha a Alonso Quijano, ya que si se convierte en don Quijote, el proceso será irreversible hasta que se encuentre en el lecho de muerte.

- 148.–*Don Quijote* se encuentra camino de *El Toboso* y sigue en su intento de conquistar el corazón de su amada *Dulcinea*, y qué mejor que demostrarle su valentía enderezando entuertos a lo largo y ancho de Castilla. *Sancho* le ha proporcionado un plano con diversos lugares donde poder afrontar una batalla para así obtener mayor fama. Se sabe cuál es la distancia de cada uno de esos lugares hasta *El Toboso* y la distancia de esos lugares entre sí. Cada vez que *Don Quijote* dé feliz término a una aventura, lo cual le requerirá un tiempo t_i , obtendrá una fama f_i y mandará que el vencido vaya hasta *El Toboso* para dar noticia de la victoria de *Don Quijote*. El problema es que como la distancia es el olvido, el vencido aunque cumple con su promesa va disminuyendo la fama que publica de Don Quijote en una constante proporcional k . Es decir, si d es la distancia hasta el toboso, y f la fama profesada por el vencido, entonces la fama que profesará ante *Dulcinea* disminuye en kfd . Además *Don Quijote* se ha propuesto estar en *El Toboso* antes de D días, pues no puede esperar más para encontrarse con *Dulcinea* y hacerle llegar todo el amor que siente por ella. Diseñe un algoritmo por **Backtracking** o **Programación Dinámica** que maximice la cantidad de fama obtenida y permita a *Don Quijote* estar en *El Toboso* para el día señalado. **NOTA:** Considérese que si la fama a profesar es negativa, entonces el vencido no irá a ver a *Dulcinea*.

- 149.–**Don Quijote y Sancho Panza cabalgan de nuevo.**

Don Quijote y Sancho se han decidido a emprender nuevas aventuras. Su fama ha llegado a los más recónditos lugares, sobre pasando los límites de La Mancha, y extendiéndose más allá de las tierras de Castilla y Aragón. Han estado sopesando hacer Las Américas, porque las verdaderas aventuras están donde ningún mortal puso el pie, pero han desistido porque ninguno de los dos es muy amigo de andar sobre el agua.

Continuamente les llegan mensajes desesperados de gentes que no ven solución a sus males y precisan la desinteresada ayuda de un esforzado caballero. No pueden atender a todas las peticiones puesto que no disponen de tanto tiempo como quisieran, y tampoco las fuerzas son las mismas que en sus años mozos. Pero están resueltos a desfacer el máximo número de entuertos que puedan en el tiempo de que disponen y con las fuerzas que les acompañan.

Si algo han aprendido nuestros hombres de sus experiencias pasadas es que necesitan saber donde ir y que las aventuras que emprendan no deben sobrepasar sus fuerzas. Por ello, se han provisto de un mapa en el que han anotado datos que consideran esenciales para culminar con éxito sus aventuras. Así por distintas averiguaciones saben el tiempo T_{ij} que tardan en ir de un lugar i a otro j , y los esfuerzos Q_{ij} de don Quijote, y S_{ij} de Sancho, que les supone ir de i a j .

Así que cargados cada uno con fuerzas F_Q , don Quijote, y F_S , Sancho, y sabiendo que disponen de un tiempo T , puesto que se han comprometido con sus más próximos para estar de vuelta en su casas y celebrar un año más el aniversario de su primera salida, *en ese lugar de La Mancha...*

Pero nuestros hombres no saben cuál es el camino que tienen que seguir para lograr su propósito de desfacer el mayor número de entuertos, según las consideraciones anteriores. Aunque ha querido el destino que el día anterior al señalado para su partida un viajero ha

llegado al *lugar de La Mancha*, con extrañas vestimentas, y algunos artilugios que de vez en cuando lanzan algún gruñido.

A cambio de un buen caldo, alguna conversación y la promesa de un techo bajo el que pasar la noche les ha proporcionado el camino a seguir, tras estar manipulando la caja que lleva consigo un rato, y lanzar entre tanto, algún que otro improperio a la misma. Improperios que a Sancho le han recordado los tiempos en que a don Quijote le sorbieron el seso los libros de Caballerías.

Interrogado el viajero por cómo sabía la solución al problema, respondió que venía de un tiempo en el que había máquinas que surcaban los cielos, y que con una caja como la que el llevaba podía hablar con otros de lejanas tierras, y resolver problemas como los de ellos tenían, con el uso de la Programación Dinámica y el Backtracking. Porque él había ido a la Universidad y seguido un curso de Ampliación de Programación, entre otras materias.

Sancho Panza al oír semejantes nombres, le ha dicho poco más o menos lo mismo que lo que le dijo a Sansón Carrasco con la gramática. Y ha empezado a sospechar que lo que quiere el viajero es comida y cama gratis. Pero don Quijote le ha aceptado sin recelos cuando les ha contado sus peripecias.

Para el viajero todo empezó cuando bajó a una cueva a rescatar unas doncellas cuyos lamentos se oían a distancia, y que en sus sollozos y lamentos, suplicaban las liberaran de su cautiverio, y al salir de la cueva no encontró sino lagunas en las que el agua hacía mucho ruido, que bien pudiera confundirse con los lamentos de las doncellas.

Los ojos de don Quijote comenzaron a brillar, por fin había encontrado evidencia de lo que en tiempos pasados pensó, y le dijo a Sancho. Y alabando la determinación del viajero, para socorrer gente en apuros, para lo que recordó lo que en cierta ocasión le dijo a Sancho:

“La libertad, Sancho, es uno de los más preciosos dones que a los hombres dieron los cielos; con ella no pueden igualarse los tesoros que encierra la tierra ni el mar encubre; por la libertad, así como por la honra, se puede y debe aventurar la vida, y, por el contrario, el cautiverio es el mayor mal que puede venir a los hombres.”

Don Quijote está resuelto a no perder un momento y partir a desfacer el mayor número de entuertos en el tiempo T de que dispone.

También es cierto que el viajero continuamente les preguntaba a nuestros hombres si eran reales, y no fruto de algún sueño o alucinación que a veces tenía resolviendo problemas de programación.

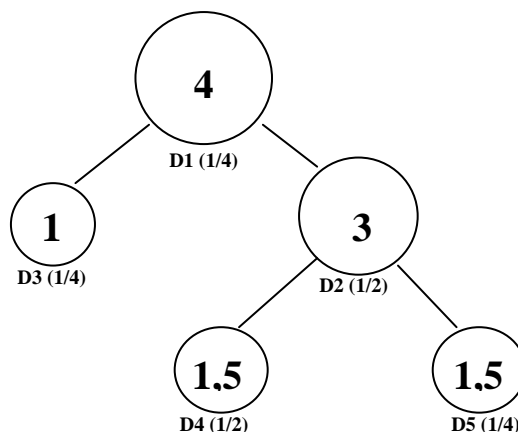
Si vos hubiérais sido el viajero ¿qué programa, basado en **Programación Dinámica** o **Backtracking**, habríais creado para ayudar a don Quijote y Sancho?

- 150.—En el juego del **Pang** tenemos a un explorador que tiene que usar una pistola con diferentes tipos de disparo para eliminar gradualmente B bolas de diferentes volúmenes V_1, V_2, \dots, V_B . Cuando el disparo del jugador impacta contra una bola, dicha bola se divide en dos bolas de menor volumen. Existen n tipos de disparos que se van realizando de forma consecutiva y circular, de forma que cuando se ha realizado el n -ésimo disparo, el siguiente será igual al primero de la secuencia. Cada disparo divide la bola en 2 bolas repartiéndose el volumen

entre las dos bolas según el tipo de disparo. La secuencia de disparos es la siguiente: $\{p_1, p_2, \dots, p_n\}$, lo que quiere decir que para el disparo k comprendido entre 1 y n una bola de volumen V se divide en 2 bolas de volúmenes $V_1 = V * p_k$ y $V_2 = V - V_1$. Cuando las bolas tienen el tamaño menor o igual a V_{\min} , si vuelven a ser disparadas, desaparecen.

El problema consiste en **encontrar el mínimo número de disparos necesarios para conseguir eliminar todas las bolas**.

Por ejemplo: Tenemos una bola ($B=1$) de volumen $V_1=4$, siendo $V_{\min} = 2$, y tenemos 2 tipos de disparos: $\{1/4, 1/2\}$. La mejor solución serían 5 disparos, como muestra la figura.



Sin embargo, si el disparo 2 hubiera sido a la bola de volumen 1, habríamos tenido que hacer al menos 7 disparos (ya que la bola de volumen 3 se dividiría en 2 bolas de volúmenes 0,75 y 2,25 respectivamente).

Diseñe e implemente un algoritmo basado en **Programación Dinámica** o **Backtracking** que resuelva el problema del **Pang**.

- 151.—La compañía *Timofónica* ha sido la adjudicataria de ubicar cabinas telefónicas en cada uno de los Campus de la Universidad. Centrándonos en el Campus de Ciudad Real, se han determinado n posibles puntos de ubicación para las cabinas. Se conocen las distancias entre ellos y el coste de instalar una cabina. Concretamente, d_{ij} es la distancia entre las ubicaciones i y j , y c_i es el coste de instalar una cabina en el punto i .

El contrato establece que desde cualquier punto (de los n identificados) hasta la cabina más próxima no puede haber una distancia superior a D . A *Timofónica* le interesa hallar las ubicaciones en las que colocar cabinas de manera **que se cumplan las condiciones del contrato**, pero que **minimice los costes de instalación**, pues no confía en la rentabilidad de la operación, que sólo aceptó por una cuestión de imagen. Diseñe un algoritmo que resuelva el problema.

- 152.—La empresa *Enredando Inc.* desea contar con las últimas tecnologías y ahora se ha embarcado en el proyecto de dotar a todas sus instalaciones con red inalámbrica (WiFi). La responsabilidad de la implantación de la red WiFi ha recaído en su departamento técnico, que ha realizado un estudio al respecto. En el estudio se incluye un mapa de las instalaciones de la

empresa en el que se muestra la disposición de los puntos en los que se pueden instalar las antenas, los puntos de la empresa que deben tener cobertura y la distancia [lineal] existente entre cada par de puntos.

Las especificaciones técnicas de las antenas WiFi indican que tienen una cobertura circular de R metros de radio y pueden ser configuradas en tres canales diferentes. Además, advierten que en el caso de que exista un solapamiento en la cobertura de las redes, este solapamiento no debe ser de señales pertenecientes al mismo canal, pues se producirá una interferencia en las señales y la zona de intersección quedará sin cobertura.

Diseñe un algoritmo basado en *Programación Dinámica* o *Backtracking* que encuentre la disposición óptima de las antenas (posición y canal), de tal manera que utilizando el menor número de antenas se proporcione cobertura a todos los puntos especificados, evitando las interferencias.

153.—La empresa de instalaciones eléctricas “*Los Bombillas*”, ha resultado la adjudicataria del concurso para la instalación del alumbrado de la ciudad. Han realizado un estudio y disponen de un plano en el que se detallan los puntos que deben iluminarse, y que son los mismos en los que es posible instalar farolas, además tenemos la distancia lineal entre ellos, y se sabe que cada farola tiene un radio de iluminación de R metros. Se trata de **iluminar todos los puntos señalados minimizando el número de farolas empleadas**. Diseñe un algoritmo *Backtracking* que resuelva el problema.

154.—*Fotoman* es un fotógrafo *freelance*, que cobra según los eventos que cubre. En la ciudad hay N eventos que pueden ser cubiertos. Para cada evento sabe el beneficio que le reporta el cubrirlo, el instante del inicio $t_{e, inicio}$ y del final $t_{e, fin}$ del evento, y del tiempo t_e que tarda en cubrirlo ($t_e \leq t_{e, fin} - t_{e, inicio}$). Dispone también de la distancia en tiempo que tarda en desplazarse entre cada dos eventos t_{ij} . Diseñe un algoritmo basado en *Programación Dinámica* o *Backtracking* que permita a *Fotoman* **maximizar el beneficio de los eventos cubiertos**.

155.—*De Campaña*. Tenemos un mapa de carreteras con N puntos críticos, que podemos clasificar en *Puntos Negros* y/o *Recaudatorios*. Dado un periodo determinado de tiempo fijo, para cada punto sabemos la recaudación que se obtiene y el número de accidentes que se producen atendiendo a estas tres alternativas: (1) no cubrirlo; (2) cubrirlo con un radar; (3) cubrirlo con una dotación de agentes de tráfico. Disponemos de R radares y P dotaciones de agentes de tráfico ($P < R$). Se pide resolver uno y sólo uno de los dos siguientes problemas:

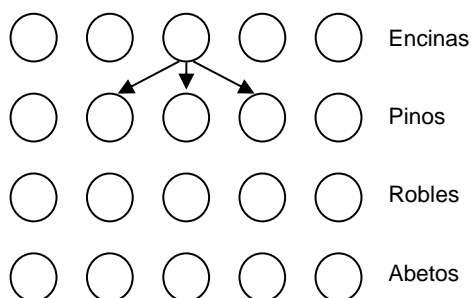
- a) Maximizar la recaudación sin que se sobrepase una cantidad M de accidentes prefijada de antemano.
- b) Minimizar el número de accidentes.

Diseñe el algoritmo que resuelva la alternativa elegida.

- 156.–El Tío Antonio tiene en su granja dos vacas: *Devoradora* y *Listilla*. Antes de ordeñarlas las alimenta llenando una hilera de N cubos con pienso (N es par). Cada cubo i contiene una cantidad p_i de pienso indicada en el cubo (todas las cantidades son distintas). Cada vaca en su turno debe elegir el 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 a priori por un procedimiento cualquiera. El objetivo de ambas vacas es comer en total lo máximo posible. La estrategia de *Devoradora* consiste en pensar poco y escoger el cubo de los extremos que esté más lleno. En cambio, *Listilla* prefiere pensárselo un poco más, para lo que ha adquirido lo último en computadoras vacunas portátiles.

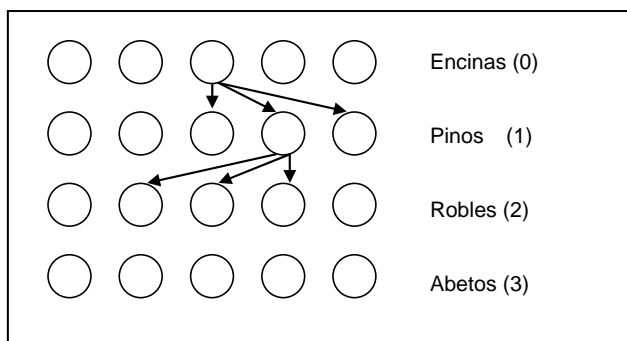
Listilla ha cursado un *master en Informática por la Escuela Superior Bovina*, pero en dicha escuela no estudian la **Programación Dinámica**, por lo que solicita su ayuda para diseñar un algoritmo utilizando dicha técnica, suponiendo que le toca empezar a escoger.

- 157.–**Problema del leñador.** Un leñador trabaja en una plantación en forma de cuadrícula de $M \times N$ árboles. Cada fila del bosque está formada por M árboles del mismo tipo, y el bosque tiene N filas de árboles de diferente tipo. Cada árbol tiene una anchura propia A que determina el tiempo que tarda en ser talado. Un leñador tiene que talar N árboles todos de tipos diferentes. Para ello, el leñador va talando el bosque de fila en fila, de forma ordenada. El leñador sólo puede talar los árboles de la fila siguiente que están o bien en la vertical o en la diagonal con respecto al árbol que acaba de talar. Véase la figura adjunta. El problema consiste en encontrar una solución que minimice el tiempo que tarda el leñador en talar N árboles, cada uno de diferente tipo.



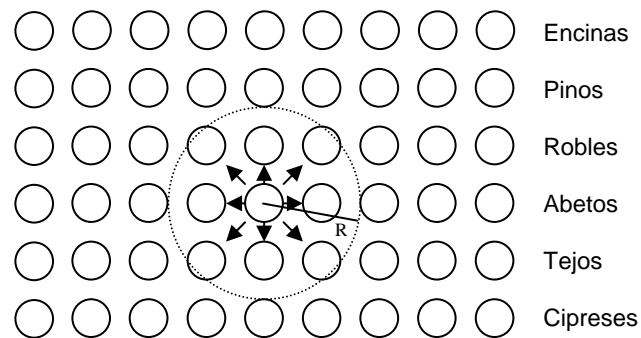
Resuelva el problema mediante estrategias de **Programación Dinámica** o **Backtracking**.

- 158.–**Problema del Leñador v.2:** Suponga que ahora en las filas pares el leñador puede talar el árbol de la fila siguiente que está en la vertical y los dos a su derecha, y en las filas impares el árbol de la fila siguiente en la vertical y los dos a su izquierda. Véase la siguiente figura:



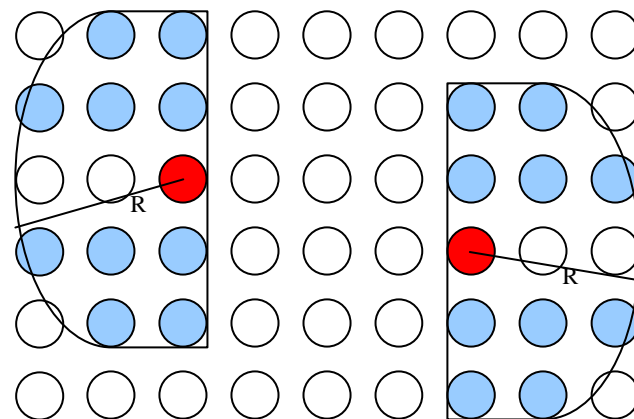
Resuelva el problema mediante estrategias de **Programación Dinámica** o **Backtracking**.

- 159.–**Problema del Leñador Variante:** El leñador puede talar cualquier árbol que esté en un radio R con respecto al que acaba de talar. Para ello, los árboles están separados una distancia constante entre sí. Véase la figura que sigue. El problema consiste en encontrar una solución que minimice el tiempo que tarda el leñador en talar N árboles, cada uno de diferente tipo.



Diseñe un algoritmo **Backtracking** que resuelva esta variante.

- 160.–**Problema del Leñador Variante 2:** Si la fila en la que acaba de talar un árbol es par, entonces los siguientes árboles que puede talar son los que estén dentro del radio y en la misma columna o columnas a la izquierda. Si la fila es impar, solo puede talar los árboles que estén dentro del radio y en la misma columna o columnas a la derecha. Véase la siguiente figura:



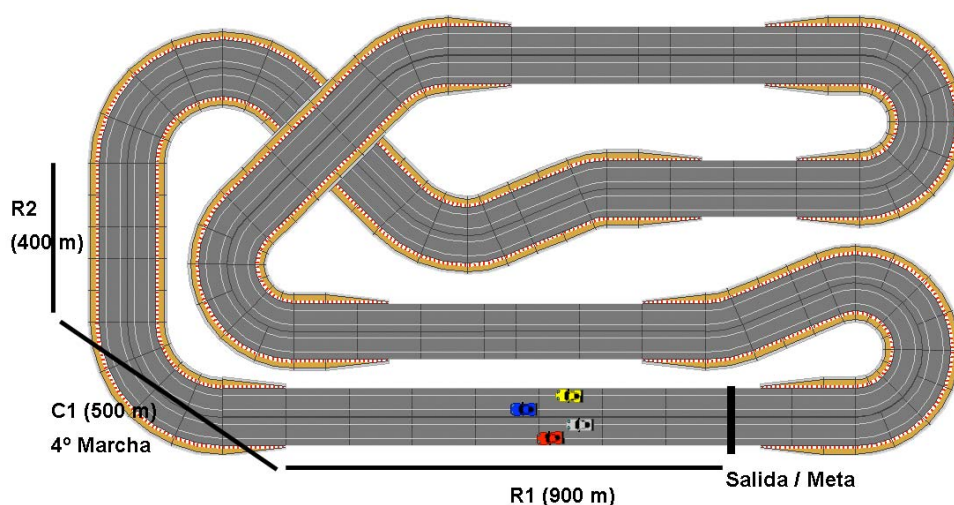
Resuelva esta variante del problema del leñador planteada utilizando **Backtracking**.

- 161.–**Problema de los Remontes.** Un esquiador quiere subir hasta la cima, ubicada a M metros de altitud. Para ello debe coger una combinación de varios remontes partiendo de altitud 0. La estación está organizada en L niveles distanciados unos de otros en N metros de altitud ($N = M / L$). De cada nivel salen R remontes. Cada remonte sube una cierta altura en un tiempo

concreto según una tabla conocida. **El problema consiste en encontrar la ruta que minimice el tiempo que tardará el esquiador en subir los M metros.** Diseñe un algoritmo basado en *Programación Dinámica* o *Backtracking* que resuelva el problema.

- 162.—**Fórmula 1.** La nueva normativa de la Federación Internacional de Automovilismo obliga a los equipos de F1 a reducir considerablemente su presupuesto. El equipo Ferrari se ha visto obligado a prescindir de su simulador de carreras y se pone en manos de los estudiantes de Ampliación de Programación.

Disponemos de la representación de un circuito. Se trata de una consecución alternativa de rectas R_i y curvas C_j , conociéndose los metros que tienen. Además conocemos que un monoplaza recorre un número de metros establecido por unidad de tiempo dependiendo de la marcha M que lleve. Suponiéndose seis posibles marchas (M_1, M_2, \dots, M_6), siempre se cumple que si mr_i son los metros recorridos con la marcha i por unidad de tiempo, tenemos $mr_1 < mr_2 < \dots < mr_6$. Finalmente, conocemos la marcha máxima con la que se puede llegar a una curva para no salirse de la misma. En cada unidad de tiempo sólo se puede subir una marcha, bajar una marcha o mantener la marcha actual. Se desea saber cuál es la secuencia de marchas que debe poner Alonso para minimizar las unidades de tiempo que tarda en recorrer una vuelta del circuito, sin salirse en ninguna curva. Diseñe un algoritmo basado en *Programación Dinámica* o *Backtracking* que resuelva el problema.



- 163.—La **Liga Fabulosa** es una competición en la que los participantes disponen de un presupuesto P para configurar un equipo de fútbol a partir de un listado dado de jugadores. De cada jugador se conoce la posición en la que puede jugar, que puede ser Portero, Defensa, Centrocampista o Delantero. El equipo debe tener exactamente once jugadores, respetando que debe haber un portero y un mínimo de dos y un máximo de cinco jugadores de cada una de las posiciones restantes. Para cada jugador j , se conoce el precio p_j de su ficha, y su valoración v_j ($v_j \in [0, 10]$) en base a sus resultados en las ligas anteriores. Diseñe un algoritmo basado en *Programación Dinámica* o *Backtracking* que determine los once jugadores que deben ficharse para conformar el equipo, sin sobrepasar un presupuesto dado P y

maximizando la valoración total del equipo, es decir, la suma de las valoraciones de los jugadores del mismo.

164.–Pedro, ya mayor, ha venido a Valladolid desde Ávila, también lo han hecho Paco y Régula desde el cortijo, y con Cipriano Salcedo se han dirigido a ver a su admirado maestro Miguel. Durante la conversación han comentado que en *El Norte de Castilla* ha aparecido una crónica relativa a un curioso caso: un sujeto anónimo ha pensado en regalar a sus amigos libros de don Miguel. Para cada libro sabe su precio y el placer o preferencia que cada uno de sus amigos tiene por cada libro. Tiene N libros y M amigos ($M \leq N$), se trata de **maximizar el placer total según los libros regalados, regalando un libro a cada uno**, si bien también sabe que sus amigos tienen algunos de los libros, y por tanto, el regalarle uno de esos libros a alguien que lo ya lo tiene, no le proporciona ningún placer, aunque le estará agradecido. Diseñe un algoritmo que resuelva el problema utilizando **Backtracking** o **Programación Dinámica**.

165.–En la comarca de “*El Buen Queso*”, cada pueblo de los N existentes, se ha especializado en fabricar un tipo de queso de los M tipos por los que es conocida. Además en cada pueblo se vende el queso a un determinado precio. Pepe Carachanca, conocido como “*El Quesero*”, debe de hacer un reparto para la gran ciudad de P tipos de quesos distintos ($P \leq N$) que le han encargado sus clientes. Dispone de un mapa con la distancia en tiempo de un pueblo a otro de la comarca, y el tiempo que tarda en un pueblo en comprar el queso, diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que permita a Pepe, conseguir recoger los P quesos en el menor tiempo posible.

Suponga ahora que Pepe dispone únicamente de un tiempo T , sabe el precio al que le cuesta cada tipo de queso (el mismo todo el pueblo), y el precio al que se lo vende a su cliente, y trata de conseguir maximizar su beneficio. Diseñe un algoritmo que resuelva el problema.

166.–**Los Mayos.** Comienza el mes de mayo, y una cuadrilla de mozos se dispone ha efectuar la ronda de los mayos, a las mozas del pueblo. El *Informático* ha venido al pueblo a tomar fuerzas para enfrentarse a los exámenes finales, y le han encargado la tarea de conseguir rondar de forma “efectiva” a todas las mozas. Ha recopilado algunos datos como el tiempo de desplazamiento entre las casas de las mozas a rondar y de cada moza el tiempo que tardan en cantarle el mayo. Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que devuelva para un tiempo dado, cuál ha de ser la ruta para rondar a la mayor cantidad de mozas de una lista dada.

MODIFICACIÓN: Resuelva el problema teniendo en cuenta que al *Informático* algunos de los mozos de la cuadrilla lo han sobornado para que necesariamente ronden a una serie de mozas.

167.–Juanito se ha comprado una cosechadora para vendimiar. Se aproxima la época de la recolección de la misma y a Juanito le llegan muchos clientes para pedirle que recoja su

cultivo ya que éste tiene unos precios muy competitivos. Ante la avalancha de clientes, Juanito tiene que hacer una selección de los mismos ya que le va a ser imposible atender a todos porque la uva tiene una fecha a partir de la cual pierde su valor. De cada parcela se sabe precio que tiene que cobrar, el tiempo que emplea en la recolección y la fecha a partir de la cual no merece la pena recoger el producto porque “caduca” y no se producen beneficios.

Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que ayude a Juanito a maximizar el beneficio puesto que necesita amortizar cuanto antes el coste de la cosechadora.

Resuelva el problema suponiendo que un cliente puede tener varias parcelas, y que todas las parcelas del cliente tienen que recolectarse consecutivamente.

168.—El presidente de la sociedad ha encargado una encuesta a una agencia de sondeos afín (tiene estrechos lazos con el presidente de la agencia puesto que lo nombró él). Pretende que la encuesta avale su gestión y sirva para desprestigiar a la competencia. La agencia ha parcelado el dominio en N zonas. Por sondeos anteriores, de cada zona se sabe el porcentaje de partidarios del presidente y de los partidarios de la competencia. Se trata de elegir M zonas de entre esas N de tal forma que la diferencia entre partidarios del presidente y de los de la competencia, se maximice. Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que resuelva el problema.

169.—Una potencia invasora ha planificado la invasión de una serie de países, cada invasión supone un coste en descontento d_i entre su propias tropas y sus ciudadanos (cuesta vidas e impuestos), y un beneficio b_i , puesto que puede disponer de sus materias primas. Además el país ofrece una resistencia a la invasión r_i . El departamento de planificación también ha establecido un orden entre algunos países para ser invadidos, representado en una secuencia, s . Así, si $i < j$, entonces el país s_i debe invadirse antes (no necesariamente inmediatamente antes) que el s_j , puesto que sino es inviable el éxito. Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que resuelva el problema.

170.—Un barco de transporte de mercancías, con una capacidad máxima de carga C , ha sido adquirido por una empresa que quiere maximizar los beneficios obtenidos con su utilización. El barco transporta mercancías entre una lista de P puertos. En cada puerto se pueden comprar y vender M productos diferentes. El precio de producto varía de un puerto a otro. En cada puerto lo que hace es vender el producto que lleva de carga (vende toda su carga) y carga otro producto de ese puerto (la cantidad a cargar dependerá de la capacidad del barco, y de la disponibilidad de capital). Conocemos los días que tarda el barco en ir desde cada puerto a todos los demás (se incluye el tiempo de carga y descarga). El capitán dispone de un capital inicial D para comprar mercancías.

El problema consiste en encontrar la ruta que maximiza los beneficios en un tiempo T (en días), teniendo en cuenta que la ruta debe comenzar y terminar en el mismo puerto, y además, no se puede repetir ningún puerto. Diseñe un algoritmo basado en **Programación Dinámica** o **Backtracking** que resuelva el problema.

- 171.–**Ascensores Inteligentes.** Un edificio de P plantas dispone de dos ascensores, cada uno con una capacidad de C personas, que tardan un tiempo de t segundos en desplazarse de una planta a la siguiente. Se ha detectado que la hora punta es entre las 11:00 y las 11:05 habiendo un tránsito masivo de personas que se desplazan a través de las distintas plantas. Cada persona que desea hacer un trayecto, indica la hora a la que desea coger un ascensor, la planta origen y la planta destino.

Supuesto que tenemos la lista de N trayectos solicitados, correspondientes a otras tantas personas (esto es, una persona solo solicita un trayecto), se trata de planificar el funcionamiento de los ascensores para que se minimice el tiempo de espera para entrar en el ascensor del conjunto de las personas. Diseña un algoritmo basado en **Programación Dinámica** o **Backtracking** que resuelva el problema.

Resuelva el problema si se trata de minimizar el tiempo de servir las N peticiones, sin que una persona sobrepase un cierto tiempo de espera $TEsperaMaximo$ antes de subir al ascensor, ni esté en el ascensor más de un tiempo $TDentroMaximo$.

Nota: Despréciase el tiempo de apertura y cerrado de puertas, y el tiempo empleado en entrar y salir del ascensor.

- 172.–Tras conseguir proclamarse campeones del mundo, la selección de baloncesto ha regresado a casa. “*Las Autoridades*” han pensado en homenajearles por todo lo alto. Conscientes de que cada homenaje le proporciona al equipo nacional en su conjunto (dirección técnica incluida) una satisfacción dada, que suele ser distinta según el homenaje, quieren maximizar la satisfacción total de la selección nacional en los homenajes que les hagan, como reconocimiento.

Puestos a pensar han elegido N homenajes posibles, de cada uno de los cuales saben su duración d_i y la satisfacción s_i que les proporciona. Además, los homenajes se van a realizar en distintos lugares, y también se sabe el tiempo de desplazamiento t_{ij} entre los lugares en que se desarrollarán los homenajes i y j . El instante en que comienza el homenaje es indiferente, puesto que tienen carta blanca para detener el tráfico en la Gran Vía o el Paseo de la Castellana, o incluso suspender la sesión parlamentaria del día –si es que a esas horas sus señorías se encontraran trabajando o debatiendo...

Hay un homenaje fijo, que es la recepción (“*La Recepción*”) a la selección por parte de *Las Autoridades*. Dada la apretada agenda de las mismas, con compromisos ineludibles, “*La Recepción*” debe comenzar a las C horas y finalizar a las F horas, exactamente.

El viaje de la selección ha sido largo y se merecen un bien ganado descanso. Así pues, los homenajes comenzarán por la mañana a partir de la hora *HoraInicio* y terminarán por la tarde a la hora *HoraFinal*, puesto que desean regresar con los suyos y disfrutar de unas breves pero merecidas vacaciones.

Diseña un algoritmo basado en **Programación Dinámica** o **Backtracking** que resuelva el problema, es decir, devuelva la secuencia de homenajes que maximice la satisfacción del equipo nacional y la satisfacción obtenida.

Nótese que puesto que hay un homenaje fijo, que es “*La Recepción*”, se trata de determinar los homenajes que deben realizarse de tal forma que se maximice la satisfacción de los

componentes del equipo nacional, supuesto de que disponemos de *TiempoAntes* ($= C - HoraInicio$) antes de “*La Recepción*” y *TiempoDespués* ($= HoraFinal - F$) después de la misma.

173.—En lo profundo del bosque, muchos de ellos lejos de su casa, se agolpan los *hobbits* en la entrada de la cueva para mantener distraída a *La Bestia* con sus acertijos. Reunidos previamente los *hobbits* se han aprendido todos N acertijos que sabían individualmente. Cada *hobbit* h tarda un tiempo p_{ha} en plantear el acertijo a , y *La Bestia* tarda en resolvérselo r_{ha} , no necesariamente los mismos para cada *hobbit*, puesto que cada uno tiene una forma de expresarse, y de acuerdo a como se exprese la bestia tarda más en entenderlo, aún siendo el mismo acertijo.

Diseñe un programa que indique a cada *hobbit* cuál es el acertijo que debe plantear para mantener a *La Bestia* el mayor tiempo ocupada, bajo los siguientes supuestos:

- b) A la entrada de la cueva se agolpan N *hobbits* y cada uno plantea un acertijo distinto.
- c) En la entrada de la cueva hay M *hobbits* ($M < N$), plantean todos los acertijos, cada uno distinto, pero puede haber *hobbits* que no planteen ninguno.
- d) Como en b) pero cada *hobbit* plantea al menos un acertijo.
- e) Como en c) pero ahora el tiempo de respuesta de *La Bestia* a los segundos y sucesivos acertijos planteados por un mismo *hobbit* se reduce a la mitad para cada uno respecto al tiempo considerado inicialmente.

174.—Suponga ahora que *La Bestia* tiene C cabezas ($C \ll M$), con cada una de las cuales atiende a un *hobbit*, y que en cuanto una cabeza no esté ocupada se dará cuenta de la estrategia de distracción de los *hobbits*. Cada uno de los M *hobbits* plantea un acertijo distinto, y aunque pueden ser varios acertijos, no puede plantearlos al mismo tiempo a distintas cabezas. El tiempo sólo depende del acertijo y del *hobbit* que lo plantea, no de la cabeza. El problema es el mismo que en el problema anterior, pero las hipótesis son:

- a) Hasta que no se ha resuelto completamente el acertijo, el *hobbit* no puede plantear otro acertijo a otra cabeza.
- b) El acertijo se plantea y a continuación el *hobbit* puede plantearle otro a otra cabeza.

175.—**El vicioso empedernido.**

Juan es un padre y esposo ejemplar. Su suegra espera impaciente el día 4 de mayo de cada año, fecha en la que ésta celebra su cumpleaños pues, además de hacerle un estupendo regalo, Juan, al felicitarla, llora inconsolable al teléfono por no poder estar a su lado y con el resto de su familia que van a pasar tan señalado día a casa de la abuela. Pero lo que no sabe la buena señora, ni la esposa ni los hijos pueden imaginar es que la tristeza le dura a Juan lo que tarda en colgar el teléfono ya que dispone de 24 horas para hacer lo que le venga en gana, tiempo en el que se dedica a sus vicios: **beber, fumar y jugar**.

Pero Juan es una persona metódica y "juiciosa", y ha comprobado tras un exhaustivo análisis que cada uno de sus vicios v le consume un determinado tiempo t_v y le proporciona un determinado grado de placer p_v . No obstante, también ha observado las siguientes consideraciones:

- a) Se emborracha si la razón entre las copas bebidas y el tiempo en horas que tarda en bebérselas es mayor que k , siempre que se haya bebido al menos C copas en las h últimas horas.
- b) Cuando se emborracha, en la primera hora experimenta un placer de p_b , y el resto del día duerme la mona.
- c) Borracho cualquier otro vicio no le proporciona placer.
- d) No puede fumar y beber a la vez.
- e) Si durante una partida fuma o bebe el placer proporcionado por este segundo vicio se reduce al 50%.
- f) La tercera partida que juegue seguida no le proporciona placer alguno.

¿Cuál es la secuencia que le proporciona mayor placer al cabo de las 24 horas de desenfreno de las que dispone?

176.—Proponga un problema que pueda ser resuelto por Backtracking o Programación Dinámica e impleméntelo.