UNIVERSIDAD DE JAÉN

DISEÑO DE ALGORITMOS

Grado en Ingeniería Informática
Departamento de Informática

Vuelta Atrás

Problema 1: Formas de sacar un valor dado con n dados

Se dispone de **n** dados de 6 caras. Se desea conocer el **número de combinaciones** que existen para conseguir una puntuación **m** dada con los n dados. Por ejemplo, si n=5, para sacar 6 (m=6), existen 5 formas:

1, 1, 1, 1, 2
1, 1, 1, 2, 1
1, 1, 2, 1, 1
1, 2, 1, 1, 1
2, 1, 1, 1, 1

Guía: una combinación parcial no es factible si ya es imposible obtener el valor o si ya es seguro que se pasa. Si k es el número de dados ya tirados y n-k los que faltan por tirar:

$$\left(\sum_{i=1}^{k} d_i\right) + (n-k) * 6 < m \ OR \ \left(\sum_{i=1}^{k} d_i\right) + (n-k) > m$$

Es decir, con el número de dados que lleva, no debe de estar obligado a pasarse ni obligado a quedarse corto. por ejemplo, si n=5:

- Con m=6, si lleva 4 dados (1, 1, 1, 3) o bien (1, 1, 1, 6) ya no es factible
- Con m=18, si lleva 3 dados (1, 1, 3) no es factible

Problema 2: Subconjuntos de suma exacta

2.1.- Soluciones de longitud fija.

Disponemos de un conjunto A de n números enteros almacenados en un vector. Dados dos valores enteros, m y C, donde m<n, se desea resolver el problema de encontrar todos los subconjuntos de A, compuestos por exactamente m elementos, tales que la suma de los valores de esos m elementos sea igual a C.

Por ejemplo, considérese el conjunto {3, 4, 6, 7}. Se desea encontrar todos los subconjuntos compuestos por exactamente 2 elementos, cuya suma sea igual a 10. Utilizando la nomenclatura anteriormente definida, tenemos que: A= {3, 4, 6, 7}; m= 2; n=4 y C=10, siendo posible encontrar dos subconjuntos en A, de 2 elementos cada uno, cuya suma es 10: {3,7} y {6,4}.

Se pide diseñar un algoritmo, siguiendo el esquema de vuelta atrás, que resuelva este problema.

2.1.- Soluciones de longitud variable.

Dado un conjunto de números enteros X= $\{x_1, x_2, ..., x_n\}$, encontrar todos los subconjuntos que sumen una cantidad C. Ejemplo: si X= {1, 2, 3, 4, 5} las diferentes formas de sumar 6 son {1, 2, 3}, {2, 4}, {1, 5}.

Problema 3: Combinaciones con 5 vocales

Un documento secreto está codificado con el abecedario de 8 símbolos Σ = {1, 2, 3, A, E, I, O, U}. Se sabe que en él se encuentra encriptada la información buscada en palabras de longitud no mayor de 5 que contienen exactamente 2 vocales.

Escribe un algoritmo de vuelta Atrás que genere el listado del vocabulario, es decir, de todas las palabras que podrían encontrarse en dicho documento.

Guía: Debe observarse que la palabra "AE" es solución. También lo son "AE1", "AE123" y "AE232".

Problema 4: La salida más corta de un laberinto

Un laberinto se puede representar mediante una matriz bidimensional en la que cada casilla puede estar marcada mediante un 0, un 1, un 2, un 3 ó un 4: 0 para indicar que la casilla está libre, 1 para indicar que la casilla es un obstáculo, 2 para indicar que ya se ha pasado por esa casilla, 3 para indicar la entrada al laberinto y 4 para indicar que la casilla representa la salida.

Diseñar un algoritmo basado en la técnica de Vuelta Atrás que, dada una matriz que representa un laberinto, devuelva el **recorrido más corto** (camino con menos casillas) desde la entrada hasta la salida.

Problema 5: El cambio de moneda

Sea un conjunto de monedas M = {1, 2, 5, 10, 20, 50, 100, 200} en el que hay 5 monedas de cada clase. Diseñar un algoritmo de vuelta atrás que dé el cambio con el menor número de monedas posible si la cantidad a devolver es siempre menor o igual que 1.000.

Problema 6: El camino hamiltoniano en un grafo no dirigido

En un grafo no dirigido, un camino se dice Hamiltoniano si pasa exactamente una vez por cada uno de los nodos del grafo, pero sin volver al punto de salida.

Dado un grafo no dirigido completo cualquiera, implementar un algoritmo para encontrar el camino hamiltoniano mínimo.

Guía: Usa una matriz de adyacencias para describir el grafo y explora todas las permutaciones de los *n* nodos del grafo. Cuando se encuentre una solución, su valor debe servir para podar nodos del árbol.

Problema 7: Recorrido de un Rey por el tablero de ajedrez

Dado un tablero de ajedrez de tamaño nxn, un rey es colocado en una casilla arbitraria de coordenadas (x_0,y_0). Sabiendo que en el juego del ajedrez el rey solamente se puede mover a una de las ocho casillas adyacentes, si se desea que el rey recorra todo el tablero visitando solamente una vez cada casilla, se necesitan n^2-1 movimientos.

Se le va a asignar a cada casilla del tablero un peso (dado por el producto de sus coordenadas). Por tanto a cada posible recorrido también se le puede asignar un valor que viene dado por la suma de los pesos de las casillas visitadas multiplicado por el índice del movimiento que nos llevó a esa casilla dentro del recorrido.

Esto es, si (x_0,y_0) es la casilla inicial y el recorrido R viene dado por los movimientos $[(x_1,y_1), (x_2,y_2), ..., (x_k,y_k)]$, con $k = n^2 - 1$, el peso asignado a R vendrá dado por la expresión:

$$P(R) = \sum_{i=1}^{k} (i * x_i * y_i)$$

Por ejemplo, para un tablero de 2x2, si se sitúa el rey en la casilla (1,1), dos posibles recorridos son:

R1	1	2
1	--	***1***
2	***2***	***3***

R2	1	2
1	--	***2***
2	***3***	***1***

Para estos ejemplos los valores de los recorridos serían:

- R1= 1*(1*2)+2*(2*1)+3*(2*2)=18
- R2= 1*(2*2)+2*(1*2)+3*(2*1)=14

Diseñar un algoritmo basado en la técnica de Vuelta Atrás que, dada una casilla de comienzo, devuelva el recorrido de valor mínimo para un tablero de 8x8.

Problema 8: El recorrido de un caballo por el tablero de ajedrez

Implementar un algoritmo basado en la técnica de la Vuelta Atrás que, comenzando en la casilla superior izquierda (la casilla [0][0]), recorra un tablero de ajedrez usando un caballo.

Un caballo se puede mover, siempre y cuando no se salga del tablero, a 8 casillas. Los movimientos son en forma de L (dos casillas en horizontal o vertical y a continuación una en vertical u horizontal respectivamente). Es decir, si el caballo está en la casilla (i,j) se puede mover a las casillas (i-1,j-2), (i-2,j-1), (i-2,j+1), (i-1,j+2), (i+1,j+2), (i+2,j+1), (i+2,j-1) y (i+1,j-2).

Nota: Para hacerlo más simple, considere un tablero de 12x12 en el que las dos primeras y últimas filas y columnas están marcadas como exterior del tablero. En este caso, la casilla de partida es la [2][2].

Generalizar el problema para un tamaño de tablero de nxn. En este caso puede que no haya solución para algunos valores de n. (Para n=5 si existe).

Problema 9: El juego del Tresminó

El juego de ***Tresminó*** es igual que el dominó, pero las fichas son solo 10: desde la blanca doble a la tres doble. Es decir:

- Blanca-Doble, Blanca-Uno, Blanca-Dos, Blanca-Tres, Uno-Doble, Uno-Dos, Uno-Tres, Dos-Doble, Dos-Tres y Tres-Doble.

Implementa un algoritmo de **Vuelta Atrás** que imprima todas las posibles partidas que empiecen con la ficha Tres-Doble y coloquen todas las fichas.

Problema 10: El juego de cartas de las 31

El juego de las 31 usa cartas de la baraja española con las siguientes puntuaciones : 1, 2, 3, 4, 5, 6, 7, 10 (sota), 11 (caballo) y 12 (rey) con 4 palos: oros (O), copas (C), espadas (E) y bastos (B).

Diseñar un algoritmo basado en la técnica de la Vuelta Atrás que imprima en pantalla todas las posibles formas de obtener 31 puntos utilizando siempre 4 cartas y como mucho dos palos distintos en cada combinación. Ejemplos:

- Sota_O + Caballo_O + 7_C + 3_B -> Incorrecta por tener 3 palos
- Sota_O + Caballo_O + 7_C + 3_C -> Correcta

Problema 11. Asignaciones de ordenadores y becarios a los vicerrectorados.

El rector de la Universidad de Jaén ya tiene perfilado su nuevo equipo de gobierno. De los nuevos Vicerrectorados creados *n* requieren una instalación informática y personal que la dirija. El rector les ha asignado *n* máquinas y *n* alumnos que cursan la asignatura de Diseño de Algoritmos para que asuman la dirección.

Pero no todas las máquinas son apropiadas para las necesidades de todos los vicerrectorados. Se dispone de una función booleana **apropiada*(m,v)*** que indica si la máquina *m* es apropiada para el vicerrectorado *v*. Asimismo, no todos los alumnos conocen todas las máquinas, por lo que disponemos de una función booleana **conoce(*a,m)*** que indica si el alumno *a* conoce la máquina *m*. Finalmente, no todos los alumnos aceptan trabajar en todos los vicerrectorados, por lo disponemos de la función booleana **acepta*(a,v)*** que indica si el alumno *a* acepta o no trabajar en el vicerrectorado *v*.

Se pide diseñar una función basada en la técnica de vuelta atrás capaz de encontrar, si existe, una manera de asignar a cada vicerrectorado una máquina apropiada a sus necesidades, y un alumno que conozca la máquina y que acepte trabajar en el vicerrectorado, si es que tal asignación existe. En caso de que no sea posible la asignación, el programa también debe devolverlo.