Olimpiada Informática Española 2016
Soluciones y estadísticas

Conexiones aeropuertarias

Problema:

Dadas n ciudades situadas en línea y sus posiciones p_i, dos ciudades están conectadas si su distancia es mayor o igual a d. ¿Cuál es la máxima d tal que todo par de ciudades está conectado?

Solución:

Sea x la ciudad en la menor posición e y la ciudad en la mayor posición. La solución será $\min\limits_{1 \leq i \leq n} \max(p_i - x, y - p_i)$.

Código César

Problema:

Descifrar un texto en castellano cifrado con el algoritmo César.

Solución:

Una posible solución es buscar la letra más frecuente, que típicamente en castellano es la `A` o la `E`. Se puede terminar de decidir buscando palabras frecuentes o intentando minimizar letras poco frecuentes como `W`.

GPS

Problema:

Encontrar un punto en el espacio delimitado por el cubo $[0, mx - 1] \times [0, my - 1] \times [0, mz - 1]$.

Solución:

Se puede resolver usando tantas preguntas como dimensiones no fijadas, montando un sistema lineal con una ecuación para cada dimensión donde las incognitas son las coordenadas del punto buscado. Haciendo preguntas en vertices opuestos siempre se obtendrán ecuaciones independientes.

Árbol filogenético

Problema:

Dado un árbol binario completo con valores en las hojas, se debe asignar valores a los nodos internos de forma que se maximice el número de nodos que tienen el mismo valor que su padre.

Solución:

Sea $p(u, x)$ la máxima parsimonia del subárbol del nodo u si le ponemos el gen x. Se puede ver que si sus hijos son a y b, entonces

$$p(u,x) = \text{máx}(p(a,x)+1, \max_{1\leq j\leq g} p(a,j)) + \text{máx}(p(b,x)+1, \max_{1\leq j\leq g} p(b,j))$$

donde $p(a, x) + 1$ y $p(b, x) + 1$ sólo se pueden considerar si el gen x ha aparecido en cada uno de los subárboles hijos.

Comiendo setas

Problema:

Dos personas deben comer n setas en orden y tienen un beneficio A_r, B_r tras comer la seta r. Comer una seta i tras la seta j conlleva una penalización $T(i,j)$. Se busca maximizar el beneficio total.

Solución:

Sea $f(i,j)$ el máximo beneficio que se puede obtener si la última seta comida por el primero es i y la última seta comida por el segundo es j. Sea $k = \text{máx}(i,j) + 1$ la siguiente seta. Entonces,

$$f(i,j) = \text{máx}(f(k,j) + A_k - T(i,k), f(i,k) + B_k - T(j,k))$$

Saltos en el plano

Problema:

Dados n puntos en el plano decir el máximo número de puntos que podemos unir de manera que entre un punto y el siguiente no disminuya ni la coordenada x ni la y.

Solución:

Este problema se puede resolver utilizando estructuras de datos como el Segment Tree o reduciendo el problema a encontrar la subsecuencia creciente más larga.

Los puentes de Enelogonia

Problema:

Dado un grafo conexo decir si se le pueden añadir aristas no repetidas para que contenga un ciclo euleriano. Y si se puede, decir una posible solución.

Solución:

Si hacemos una búsqueda en anchura en el grafo complementario se nos genera un bosque. Se puede ver que existe solución si i sólo si hay una solución usando solamente las aristas del bosque, por lo tanto, si se puede, se puede para cualquier $k >= n - 1$. Visto esto, es fácil ver si existe una solución usando un greedy para generarla.

Palabra chupiguay

Problema:

Decir si una palabra cumple una serie de restricciones.

Solución:

La solución más sencilla consiste en aprovechar que la palabra tiene como mucho 14 letras. Basta entonces con probar todos los posibles finales para una palabra `chupi` y mirar si se cumplen las restricciones del enunciado.

El equilibrio de Nash

Problema:

Encontrar los equilibrios de Nash dado un juego para dos jugadores.

Solución:

Calculamos para el primer jugador el máximo de cada fila y para el segundo el máximo de cada columna en sus respectivas matrices. Una posición (i, j) es un equilibrio de Nash si el valor en la primera matriz coincide con el máximo de esa fila y en la segunda con el máximo de esa columna.

El laberinto del velociraptor

Problema:

Encontrar la salida de un laberinto en el que se puede avanzar o girar en ambos sentidos.

Solución:

Existen varias soluciones que pueden obtener 100 puntos. Una posible es la que conceptualmente consiste en poner la mano izquierda contra la pared e ir avanzando sin separarla de la pared.

Reparto de caramelos

Problema:

Buscar el mínimo número de caramelos que se tienen que comprar si queremos repartirlos todos como premio de manera justa en una competición de n personas independientemente del resultado de ésta.

Solución:

La solución esperada para este problema es la fórmula cerrada $n^2 - n$.

Subsecuencia de suma mínima

Problema:

Dada una secuencia de números enteros, calcular su subsecuencia consecutiva no vacía cuya suma sea más próxima a cero.

Solución:

Se calculan las sumas parciales del vector y se ordenan. Entonces basta comparar la diferencia entre cada par de sumas consecutivas y quedarse con la menor de ellas, teniendo en cuenta su inicio y final.

Número de caminos

Problema:

Encontrar el numero de caminos que cruzan de esquina a esquina una cuadrícula, avanzando sólo hacia abajo y hacia la derecha y evitando las p casillas prohibidas.

Solución:

Se calcula mediante programación dinámica la cantidad de caminos (no necesariamente válidos) desde el origen hasta cada una de las casillas prohibidas. Este número es el total de caminos que hay menos todos aquellos que pasan por una prohibida, que se obtienen recursivamente. El número total de caminos (no necesariamente válidos) entre dos casillas es un número combinatorio y se puede calcular en tiempo constante a partir de los factoriales precalculados módulo $10^9 + 7$.

Planificación (2)

Problema:

Dadas diversas tareas donde cada una tiene un valor v_i, un tiempo para completarla t_i y un tiempo final f_i máximo para ser completada, se pide encontrar el máximo valor si tenemos desde el instante 0 hasta el m.

Solución:

Es útil ver que podemos eliminar m si hacemos $f_i' = \text{mín}(f_i, m)$ para toda tarea. Sea $f(i, t)$ el máximo valor si nos quedan por tratar las tareas $1, \ldots, i$ y podemos calcular hasta el tiempo t. Entonces,

$$f(i, t) = \text{máx}(f(i-1, \text{mín}(f_{i-1}, t)), v_i + f(i-1, \text{mín}(f\,i-1, t-t_i)))$$

Pero esto sigue sin ser suficiente para obtener 100 puntos, es necesario utilizar esta programación dinámica con algunas optimizaciones.