Olimpiada Informática Española
Soluciones de los problemas

2009

Día 1

Mi reloj me habla [Implementación, Sorting]

Problema: Dadas dos horas, escribir por orden alfabético, todas las palabras que aparecen en un reloj digital entre esas dos horas.

Solución: Iteramos por todos los minutos entre las dos horas y comprobamos si esa hora es una palabra, de serlo la guardamos. Al final ordenamos la lista de palabras.

Gramaticando [Dinámica]

Problema: Dados unos conjuntos de palabras, responder preguntas del tipo: ¿cuántas palabras de tamaño n tiene el conjunto X? Los conjuntos se definen como la unión de *producciones*: la concatenación de letras y/o palabras de ciertos conjuntos (puede ser el propio conjunto).

Solución: La solución que proponemos es una dinámica, donde los estados son dp[c][l] que es el número de palabras de longitud l del conjunto c. La recursión aprovecha una función recursiva auxiliar f, que dado los números c, i, j, l, devuelve el número de palabras de l letras que se pueden formar con la i-ésima producción del c-ésimo conjunto a partir del j-ésimo conjunto que aparece en esa producción. De este modo la recursión de la dinámica sería:

Continuación Gramaticando

$$dp[c][l] = \sum_{i \in \mathcal{I}_c} f(c, i, 1, l - l_{c,i})$$

Donde $l_{c,i}$ es el número de letras fijas en la i-ésima producción del c-ésimo conjunto. Y la función f sería:

$$f(c, i, j, l) = \sum_{1 \leq k \leq l+1+j-n_{c,i}} dp[J_{c,i,j}][k] \cdot f(c, i, j+1, l-k)$$

Donde $n_{c,i}$ es el número de conjuntos que aparecen en la i-ésima producción del c-ésimo conjunto y $J_{c,i,j}$ es el j-ésimo conjunto de la i-ésima producción del c-ésimo conjunto. La idea consiste distribuir las letras no fijas en los distintos conjuntos de la producción que estamos considerando.

Manjatan [Implementación]

Problema: Dado un conjunto de carreteras verticales y horizontales en el plano y una serie de instrucciones de conducción, realizar esas instrucciones mientras sea posible e indicar la distancia total recorrida.

Solución: Simplemente debemos simular la situación. Es fundamental organizar bien la información.

Primos [Ad-hoc]

Problema: Factorizar un conjunto de n números $\leq 10^9$.

Solución: La solución consiste en realizar una criba de Eratóstenes. Calculamos todos los primos que vayamos a necesitar (todos aquellos $\leq \sqrt{10^9}$) de modo que podemos factorizar los números simplemente recorriendo la lista que hemos creado, hasta llegar a la raíz del número.

Variedad [Backtracking]

Problema: Dados los números n, x, y, hallar todas las palabras de n letras formadas por x letras 'C', y letras 'D' y $n - x - y$ letras 'E'. La única restricción adicional es que no puede haber 4 letras consecutivas en las que una de las tres letras no aparezca.

Solución: Para encontrar todas las palabras hacemos un backtracking. La función recursiva tendrá como parámetros la cantidad de 'C', 'D' y 'E' aún disponibles y las letras de los últimos 3 días.

Día 2

Robots Limpiadores [Dinámica]

Problema: Dada una cuadrícula $n \times m$, $n \leq 8, m \leq 40$ (en los casos difíciles) con posiciones prohibidas (X), hallar el mínimo número de robots que se deben colocar para poder llegar a cualquier punto de la cuadrícula. Los robots solo se pueden mover o verticalmente o bien horizontalmente, sin cruzar posiciones prohibidas. Además, las casillas no pueden ser cruzadas por dos robots distintos.

Solución:

- 30 puntos: la solución es directa.
- 60 puntos: es suficiente con hacer un backtracking $O(2^{nm})$.
- 100 puntos: hacemos una dinámica $O(m2^n)$.

Los estados son dp[j][bit] que representa el número de robots necesarios para llegar a cualquier punto de la cuadrícula a partir de la j-ésima columna, sabiendo que si bit$[i] = 1$, entonces había un robot horizontal que llegaba a la posición $(i, j-1)$. La recursión es:

Continuación Robots Limpiadores

$$dp[j][bit] = \min_{s \in S} \Big(cost(s, j, bit) + dp[j+1][bit_s] \Big)$$

Donde S representa el conjunto de todas las combinaciones de números positivos que suman n. Podemos entender esas combinaciones como los segmentos verticales en que se dividirá la columna j. Aquellos segmentos de longitud $(l) > 1$ serán robots verticales, y aquellos de $l = 1$ serán para robots horizontales. Las casillas X tendrán asociados fragmentos de $l = 1$.

cost(s,j,bit) representa en número de robots que gastaremos para esta columna. Es el número de segmentos $l > 1$+ el número de segmentos $l = 1$ que se colocan en una fila (i) (no X) donde bit[i] $= 0$ (no aprovechamos un robot horizontal de la columna anterior).

bit_s representa los robots horizontales escogidos. $bit_s[i] = 1$ si a la fila i le corresponde un fragmento de longitud 1 (y no es X)

Rastreador [Ad-hoc]

Problema: Dado un número en binario, contar el número de 0 y 1 y aplicar una fórmula. En algunos casos los números se dan en base 10.

Solución: Si el número no está en binario lo pasamos a binario. Una vez tenemos un número binario, contamos el número de 0 y 1 y aplicamos la fórmula que nos dicen en el enunciado.

Shuffler [Dinámica]

Problema: Dadas dos pilas p_0, p_1 de n_0 y n_1 números, obtener una única pila p, cogiendo en cada paso la primera carta de una de las dos pilas, minimizando: $\sum_{i=0}^{n_0+n_1-2} \max(0, p[i] - p[i+1])$

Solución: Una solución sería realizar una dinámica, donde los estados serían dp[i][j][k] que representaría el mínimo coste de obtener una pila con los elementos de p_0 a partir de la posición i y p_1 a partir de j, donde el último número escogido viene de la pila k. La recursión sería:

$$dp[i][j][k] = \min_{t \in \{0,1\}} (max(0, v_k - p_t[i]) + dp[i + (1-t)][j + t][t])$$

Donde v_k es $p_0[i-1]$ si $k = 0$ y $p_1[j-1]$ si $k = 1$. El coste sería de $O(n_1 \cdot n_2)$

Campeonato de sumo [Implementación]

Problema: Dado un conjunto de robots de distintos grupos y los resultados de combates entre ellos, di quienes ganan la 1a fase de la competición y quien gana la final (si se juegan los partidos que tocaría).

Solución: Simulamos el torneo con los resultados que nos dan.

Transformers [Hashing, Grafos]

Problema: Dada una palabra inicial, una palabra objetivo, y una serie de reglas que permiten transformar trozos de la palabra original en otras palabras, encontrar el mínimo número de pasos para llegar a la palabra objetivo desde la inicial, o decir que no es posible. Se garantiza que, de existir ese mínimo número de pasos, este será menor a 10. Además todas las palabras están formadas por las letras *a* y/o *b*.

Solución: Podemos entender el problema como un grafo, en el que las palabras son nodos y con algunas transformaciones nos podemos mover entre nodos. Por lo que podemos aplicar un BFS sobre las palabras, para hallar la mínima "distancia" entre la palabra inicial y destino. Además, para optimizar las comparaciones entre palabras y las transformaciones es buena idea pasar las palabras a números, entendiendo que estas son números en base 2 (*a* y *b* como 0 y 1).