Olimpiada Informática Española
Soluciones de los problemas

2011

Día 1

Chuck y la patada voladora (1) [Grafos]

Problema: Dada una cuadrícula con la posición inicial de Chuck, la del enemigo principal y las posiciones de enemigos adicionales, calcular el mínimo coste para acabar con el enemigo principal. El coste de desplazarse a una posición adyacente es de 1s y el de acabar con un enemigo es de 20s. Además, no puede haber k pasos consecutivos en los que no se acabe con un enemigo.

Solución: La idea de la solución es realizar una especie de Dijkstra. En este caso podemos explotar el hecho de que solo hay dos posibles pesos, 1 y 21. Por tanto, podemos usar una idea similar a un BFS. Usaremos dos colas, en una añadimos aquellos nodos que visitamos con un coste extra de 1 y en otra los de coste 21. Para saber que nodo vamos a visitar miramos cual de los nodos al frente de cada cola tiene un coste menor. Es importante no visitar posiciones donde hayamos estado habiendo acabando con un enemigo más recientemente.

iPeds [Ad-hoc, sorting]

Problema: Se fabrican productos que requieren de 4 piezas. Dados los tiempos en que llegan paquetes de una cierta cantidad de piezas, escribir los tiempos en los que se pueden fabricar productos y cuantos.

Solución: Simplemente ordenamos los paquetes por orden de llegada y mantenemos contadores para cada pieza. Iteramos por los paquetes, en cuanto tengamos más de una unidad de cada pieza podremos fabricar productos, limitados por la pieza más escasa.

Potaje de letras [Implementación]

Problema: Dada una sopa de letras de tamaño $n \times m$, hallar y marcar un conjunto de palabras.

Solución: Simplemente iteramos por cada fila, columna y diagonales, iterando a su vez por las posiciones de la palabra, viendo si la posición anterior la habíamos colocado en la casilla anterior en la dirección que estemos siguiendo. El algoritmo tendrá un coste de $O(nmk)$ donde k es el tamaño de las palabras.

Shift right [Implementación]

Problema: Dado un conjunto de líneas, cada una con un conjunto de palabras se pide que en cada palabra la última letra pase a la primera posición de la palabra, que en cada línea la última palabra pase a ser la primera y que la última línea pase a ser la primera. En todos los casos manteniendo el orden de las demas letras/palabras/líneas.

Solución: Una solución sería pasar el input a un vector de vector de palabras V. donde V[i][j] es la j-ésima palabra de la i-ésima fila. Solo tendremos que hacer pop_back() e inserts donde sea necesario.

Día 2

Árbol de rocas (2) [Dinámica]

Problema: Dada una disposición de nodos en filas, calcular el número de arboles distintos que se pueden obtener conectando los nodos de modo que no haya ningún cruce entre arcos y que las conexiones solo sean entre nodos de hileras consecutivas.

Solución: Observación: las conexiones entre dos filas son independientes. Es decir, podemos calcular el número de posibles conexiones entre cada par de filas consecutivas y multiplicar los resultados. Para hallar la solución haremos una Dinámica, donde los estados son dp$[n_1][n_2]$ que representa el número de conexiones válidas entre dos filas con n_1 y n_2 nodos. La recursión es:

$$dp[n_1][n_2] = dp[n_1 - 1][n_2] + dp[n_1][n_2 - 1]$$

Ya que los primeros nodos están conectados, y el segundo nodo de una fila puede ir o al primer nodo o al segundo de la siguiente.

Contabilidad [Implementación]

Problema: Dado un registro de cantidades de dinero en dolares, euros y yenes. Decir la cantidad total de dolares euros y yenes por separado.

Solución: sumamos por separado los dolares euros y yenes, esos son los resultados.

Chuck y la patada voladora (2) [Grafos]

Problema: Dada una cuadrícula, con la posición inicial de Chuck y las posiciones de sus enemigos, determinar cuales son los enemigos con los que podría llegar a acabar Chuck. La única restricción es que no puede haber k pasos consecutivos en los que no se acabe con un enemigo.

Solución: Una solución sería realizar un multibfs: un BFS desde cada enemigo a la vez, de modo que para cada posición sepamos la distancia a la que está el enemigo más cercano. A partir de aquí realizamos un BFS desde la posición de Chuck, donde no iremos a casillas que estén a distancia $> k/2$ de un enemigo.

IPv6 [Implementación]

Problema: Dado un conjunto de direcciones IPv6 y unas normas para comprimirlas, aplicar esas normas en cada dirección.

Solución: Seguir las instrucciones que se indica en el enunciado.

UTSL [Ad-hoc]

Problema: Dadas 9 posiciones $(1, \ldots, 9)$, sabiendo que hay dos torretas en las posiciones 0 y 10, y hasta 4 torretas más adicionales en algunas de las 9 posiciones, hallar aquellas posiciones cuya distancia a la torreta más cercana sea mínima.

Solución: Para cada posición calculamos la distancia a la torreta más cercana.