Las Torres de Hanoi — P52637_es

Olimpiada Informática Española — Final 2007 (2007)

En 1883, Edouard Lucas inventó, o tal vez reinventó, uno de los rompecabezas más populares de todos los tiempos — la Torre de Hanoi, como él le llamó — y que aún se usa hoy día como prototipo de algoritmo recursivo. De hecho, muchos libros de texto para estudiantes de ciencias de la computación, lo ponen como ejemplo para implementar en forma de programa la idea de recursividad. Estas son las reglas del juego:

- Tenemos tres postes (varillas en la versión de sobremesa), que llamaremos A, B y C.
- Tenemos n discos con un agujero en el centro. El número de discos no cambia mientras se está resolviendo el rompezabezas.
- Todos los discos son de diferente tamaño.
- Los discos están insertados inicialmente en el poste A, en orden decreciente partiendo desde el suelo. Es decir, el más grande debajo hasta llegar al más pequeño en la cima de la torre.
- El objetivo del juego es trasladar la torre entera desde el poste A hasta uno de los otros dos.
- Sólo se puede mover un disco cada vez, y tiene que ser uno de los que encuentran en la parte alta de una de las tres pilas (que se van formando en los respectivos postes durante el desarrollo del juego) a otra torre que tiene que estar vacía o tener en su parte superior un disco mayor que el que estamos moviendo. Es decir, nunca está permitido poner un disco encima de otro más pequeño.

Aunque, naturalmente, la mejor forma de iniciarse en los secretos de este rompecabezas es intentar resolverlo sobre uno de los muchos y variados modelos reales que se encuentran en las tiendas de regalos, si esto no es posible, una buena forma de captar la esencia del juego es escribir un programa que nos muestre en la pantalla un modelo del mismo y nos permita simular el movimiento de los discos. Pero no es esto lo que vamos a proponer en esta tarea. El siguiente paso sería escribir un programa que nos resuelva el rompecabezas de la manera más eficiente posible. Tampoco es éste el objetivo de esta tarea. El objetivo que se persigue es la implementación de un algoritmo eficiente para saber el estado de cada una de las tres torres de discos en cada etapa del proceso de solución del juego. Para facilitar la tarea, no pedimos el detalle de los discos que hay insertados en cada poste, sino únicamente cuántos hay después de un determinado movimiento, cuando se resulve el rompecabezas siguiendo un algoritmo concreto, que pasamos a describir.

Se conoce perfectamente, y es muy fácil de demostrar, que el número mínimo de movimientos de discos que hay que hacer para completar el cambio de la torre original a otro poste distinto es $2^n - 1$, siendo n el número de discos. Un algoritmo muy sencillo, pero que nos

permite resolver el rompecabezas en esta cantidad óptima de movimientos es el siguiente: para los movimientos impares, se coge el disco más pequeño de todos (digamos el número 1) del poste en el que está insertado y se le coloca en el siguiente dentro de la sucesión circular $ABCABC\ldots$; para los movimientos pares, se hace el único movimiento posible que no implique a dicho disco número 1.

Entrada

El fichero de entrada consiste en una serie de líneas. Cada una de ellas contiene dos números enteros n, m: n es el número de discos y m, que estará en el intervalo $[0,\ 2^n - 1]$, será el número del último movimiento realizado. El fichero termina con una línea que contiene dos ceros y no se debe procesar.

Salida

La salida consiste también en una serie de líneas, una por cada una del fichero de entrada (excepto la última). Cada una de ellas estará formada por tres números enteros, que representan el número de discos en cada uno de los postes A, B and C respectivamente, después del movimiento m, y teniendo en cuenta que dichos movimientos se han hecho de acuerdo con el algoritmo descrito.

Puntuación

- **TestA:** **20 Puntos**

  100 casos de pruebas con $n \leq 16$.

- **TestB:** **50 Puntos**

  100 casos de pruebas con $n \leq 50$. El número m puede no caber en un entero de 32 bits, por lo que será necesario utilizar enteros de 64 bits, como el tipo *long long* de C o C++.

- **TestC:** **30 Puntos**

  100 casos de pruebas con $n \leq 100$.

Ejemplo de entrada 1

```
1 0
1 1
2 0
2 1
2 2
3 0
3 1
3 2
3 3
3 4
3 5
3 6
3 7
0 0
```

Ejemplo de salida 1

```
1 0 0
0 1 0
2 0 0
1 1 0
0 1 1
3 0 0
2 1 0
1 1 1
1 0 2
0 1 2
1 1 1
1 2 0
0 3 0
```

Ejemplo de entrada 2

```
3 5
8 45
39 437370956880
0 0
```

Ejemplo de salida 2

```
1 1 1
4 2 2
17 12 10
```

Ejemplo de entrada 3

```
84 33405917778590387650216 67
73 7501655746272260576287
99 443387824463857406963274 64779
0 0
```

Ejemplo de salida 3

```
20 39 25
26 27 20
33 30 36
```

Información del problema

Autor : Miguel Revilla
Generación : 2025-05-13 11:23:24