UN EJEMPLO DE RECURSIVIDAD: HANOI

Por I. Ramos

1- INTRODUCCION

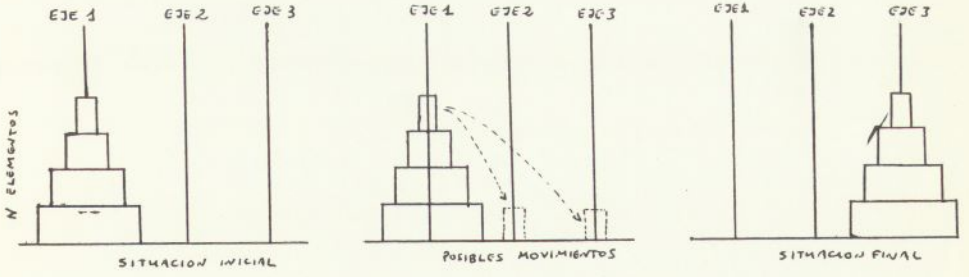
El juego de las "Torres de Hanoi" se presta a una fácil ejemplificación de los procesos recursivos. Se dice que ciertos monjes del Tíber una torre de 60 elementos de un sitio a otro y que cuando hayan terminado de trasladarla, el mundo terminará. Como veremos a continación esto no es motivo de preocupación para nadie dada la imposibilidad temporal de realizarlo.

Explicación del juego: Se dispone de N piezas circulares de radio decreciente insertadas en un eje en forma de tronco de cono. Tenemos además otros dos ejes en que insertar las piezas. El fin del juego es trasladar la torre inicial de un eje a otro teniendo en cuenta que:

1) En cada movimiento solo movemos una pieza de un eje a otro (la única accesible en la cumbre de una torre).

2) Nunca puede haber una pieza mayor sobre otra menor en un mismo eje.

Veamos una figura:

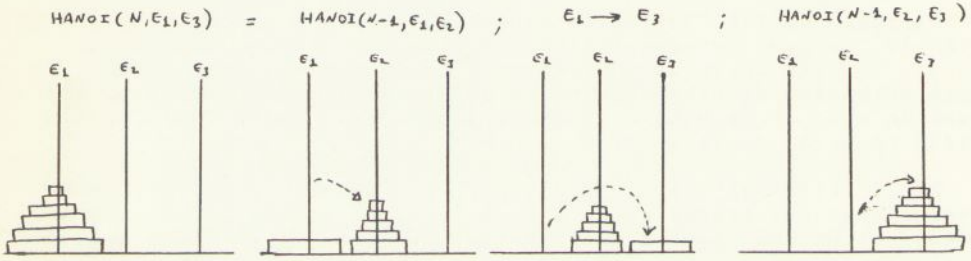


Veamos a continuación una solución trivial al problema y fácilmente implementable sobre ordenador.

2- ALGORITMO PROPUESTO

Representaremos la acción de trasladar N elementos desde un eje: E_1 a otro E_3 mediante la función de tres argumentos: HANOI (N,E_1,E_3). El algoritmo está basado en la siguiente afirmación: "El trasladar una torre de N elementos de E_1 a E_3 es equivalente a trasladar una torre

de N-1 elementos de E_1 a E_2 seguido de trasladar 1 pieza de E_1 a E_3 - y seguido de trasladar una torre de N-1 elementos de E_2 a E_3" o gráficamente:



O sea:

$$HANOI\ (N,E_1,E_3) = HANOI\ (N-1,E_1,E_2)\ .E_1 \longrightarrow E_3\ HANOI\ (N-1,E_2,E_3)$$

Si codificamos los ejes por 1,2,3 , dados dos cualesquiera de ellos: x,y (origen y fin) el intermedio será 6-x-y, con lo que la función de traslado (que llamamos HANOI) entre dos ejes cualesquiera será:

$$HANOI\ (N,x,y) = HANOI\ (N-1,x,6-x-y)\ x \longrightarrow y\ HANOI\ (N-1,6-x-y,y)$$

lo que acepta una rápida implementación en ALGOL-60 en forma de procedimiento recursivo:

```
Procedure HANOI (N,x,y);
     begin HANOI (N-1,x,6-x-y);
           WRITE (x,y);
           HANOI (N-1,6-x-y,y);
     end
```

con lo que una forma de obtener impresos los movimientos sucesivos - sería llamar a esta procedure con los parámetros adecuados

```
HANOI (4,1,3);
```

Desarrollemos este ejemplo.

```
                                                             MOVIMIENTOS
                                   | HANOI (1,1,3) .....    1-3
                   HANOI (2,1,2)   | 1⟶2 .............      1-2
                                   | HANOI (1,3,2) .....    3-2
                 |
HANOI (3,1,3)    | 1 — 3 ...........................        1-3
                 |
                                   | HANOI (1,2,1) .....    2-1
                   HANOI (2,2,3)   | 2⟶3 .............      2-3
                                   | HANOI (1,1,3) .....    1-3
```

Donde hacemos la hipótesis obvia de que HANOI (1,x,y) = x-y ya que trasladar una torre de altura 1 de x a y es realizar el movimiento de 1 pieza de x a y. Esta afirmación es equivalente a suponer que HANOI (0,x,y) es la acción nula puesto que:

```
                   | HANOI (0,x,6-x-y) ....  nada
HANOI (1,x,y)  =   | x⟶y ...............     x-y
                   | HANOI (0,6-x-y,y) ....  nada
```

que es la solución que hemos tomado nosotros en ALGOL:

```
begin  procedure  HANOI (N,x,y); value N,x,y; integer N,x,y;

              begin if N ≠ 0 then begin HANOI (N-1,x,6-x-y);
                                        ESCRIBIR (x,y,N);
                                        HANOI (N-1,6-x-y-y);
                                   end

              end

       HANOI (3,1,3) :

end
```

este programa produciría la salida dada en el ejemplo anterior.

3- IMPLEMENTACION EN ALGOL

La implementación en Algol es inmediata. Nosotros por motivos de una presentación más cuidada de la salida de la máquina, así como por obtener ejemplos para distintos valores de N hemos añadido más cosas al programa. Una versión de éste la damos a continuación

PROGRAMA

```
HAN                                                          10/26/73
                    ALCOR-ILLINOIS 7090 COMPILER

 1          'BEGIN' 'INTEGER' I,M.,'INTEGER''ARRAY' LX,LY(/1..10/).,
 2                   'INTEGER' K,MM.,
 3                   'PROCEDURE' HANOI(N,X,Y).,'INTEGER' N,X,Y.,
 4                                             'VALUE' N,X,Y.,
 5                'BEGIN' 'IF' N 'NQ' 0 'THEN'
 6             'BEGIN'          HANOI(N-1,X,6-X-Y).,
 7                              ESCRIBIR(X,Y,N).,
 8                              HANOI(N-1,6-X-Y,Y).,
 9             'END'.,
10                   'END'.,
11                   'PROCEDURE' ESCRIBIR(X,Y,N).,'INTEGER' N,X,Y.,
12                                'VALUE' N,X,Y.,
13                   'BEGIN' 'INTEGER'K.,     M.=M+1., MM.=MM+1.,
14                       'IF' MM 'EQ' 2**I-1'OR' M'EQ' 10 'THEN' 'BEGIN'
15                              LX(/M/).=X ., LY(/M/).=Y .,
16                              FORMAT(6,'('10X,10(F2.0,1H-,F2.0,3X)')').,
17                              PRINTF(LX(/1/),LY(/1/),LX(/2/),LY(/2/),
18                                     LX(/3/),LY(/3/),LX(/4/),LY(/4/),
19                                     LX(/5/),LY(/5/),LX(/6/),LY(/6/),
20                                     LX(/7/),LY(/7/),LX(/8/),LY(/8/),
21                                     LX(/9/),LY(/9/),LX(/10/),LY(/10/)).,
22                          M.=0.,
23                              'FOR' K.=1'STEP' 1 'UNTIL'10 'DO'
24                              LX(/K/).=LY(/K/).=0.,
25                                                      'END'
26                                                'ELSE''BEGIN'
27                              LX(/M/).=X ., LY(/M/).=Y .,'END'
28                   'END'.,
29                  'FOR' I.=2 'STEP' 1 'UNTIL'10 'DO' 'BEGIN'
30                  FORMAT(6,'('1H1,50X,14H*** HANOI N = ,F2.0  ,4H ***,')').,
31                  PRINTF(I).,
32                  FORMAT(6,'('///,10X,17HPUNTO DE SALIDA =,F2.0
33                                 ,18HPUNTO DE LLEGADA =,F2.0/
34                  10X,35H-----------------------------------')').,
35                  PRINTF( 1, 3).,
36                       MM.=M.=0.,  'FOR' K.=1 'STEP'1 'UNTIL' 10 'DO'
37                           LX(/K/).=LY(/K/).=0.,
38                  HANOI(I, 1, 3).,                        'END'.,
39          'END'.,
40          'FINIS'
```

Y la salida para N = 9 , E_1 = 1 , E_3 = 3 , es la siguiente:

SALIDA nº 9

*** HANOI N = 9. ***

PUNTO DE SALIDA =1.PUNTO DE LLEGADA =3.

1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	3.-1.
2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.
3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.
1.-3.	1.-2.	3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	3.-1.
2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.
3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	3.-1.	2.-1.	3.-2.
1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.
2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.
3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	3.-1.	2.-1.	2.-3.
1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	1.-3.
2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.
3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	2.-3.
1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	3.-1.
2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	2.-1.
3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.
1.-3.	1.-2.	3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	3.-1.
2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	1.-2.
3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	3.-2.
1.-3.	1.-2.	3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.
2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.
3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.
1.-3.	1.-2.	3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	3.-1.
2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	1.-2.
3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	3.-2.
1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.
2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.
3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.
1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	1.-3.
2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.
3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	1.-3.	2.-1.	2.-3.
1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	3.-1.
2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.
3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.
1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	3.-1.
2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	1.-2.
3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	3.-2.
1.-3.	1.-2.	3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	2.-1.	3.-2.	3.-1.
2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.
3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.
1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	1.-3.
2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	3.-2.	1.-3.	1.-2.
3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	2.-1.	3.-2.	3.-1.	2.-1.	2.-3.
1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	3.-1.
2.-1.	3.-2.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.	1.-3.	2.-1.
3.-2.	3.-1.	2.-1.	2.-3.	1.-3.	1.-2.	3.-2.	1.-3.	2.-1.	2.-3.
1.-3.	0.-0.	0.-0.	0.-0.	0.-0.	0.-0.	0.-0.	0.-0.	0.-0.	0.-0.

4- CONCLUSIONES

El número de movimientos que se realizan es función de N. Concretamente 2^n-1. Lo que se obtiene fácilmente sin más que considerar el árbol de generación de movimientos dado en el ejemplo. En el caso de los monjes Tibetanos el nº de movimientos es pues $2^{60}-1 \quad 10^{18}$. Es pues una leyenda tranquilizadora.

Existen muchas soluciones a este problema clásico en Teoría de Juegos, la propuesta creemos que es breve y elegante.

Debo agradecer a C. Pair el haberme sensibilizado a este problema.