make.tol de TowerHanoi.Recursive

TowerHanoi.Recursive es un programa que resuelve de forma recursiva el problema conocido como de las Torres de Hanoi. Este problema se utiliza a menudo en programación para poner en práctica la programación recursiva como sucede en el presente caso. Este programa se estructura en base a un fichero principal make.tol que contiene tanto la definicion de las variables como las funciones de resolución y funciones auxiliares, siendo la función principal de resolución HnoSolve(). El funcionamiento de este programa se ha probado para las versiones de Tol 1.1.1, 1.1.5, 1.1.6 y 2.0.1.

Las Torres de Hanói, como usualmente se le denomina en español, o Tower of Hanoi, como habitualmente se le denomina en inglés, es un juego matemático creado por el matemático francés Édouard Lucas en el año 1883. De hecho, Lucas lo comercializó como un juego, pero bajo el pseudónimo del Profesor N. Claus de Siam, mandarín del Colegio de Li-Sou-Stian que son 2 anagramas de Lucas d'Amiens (lugar de nacimiento de Lucas) y de Saint Louis. François Édouard Anatole Lucas, nacido en Amiens el 4 de abril de 1842 y fallecido en París, el 3 de octubre de 1891, además del juego de las Torres de Hanoi, trabajó en París en el observatorio astronómico y como profesor de matemáticas y destacó por sus investigaciones sobre la serie de Fibonacci y por el test de números primos que lleva su nombre, test de Lucas.

Las Torres de Hanoi, en su forma más clásica, es un juego de se basa en 8 discos, todos de diferente radio, que se pueden apilar insertándose en 1 de los 3 palos verticales que están clavados en una tabla horizontal. Estos 3 palos con sus discos insertados, forman 3 torres de discos que son las que dan nombre al juego. Partiendo de una posición inicial, con los 8 discos insertados en el primer palo y ordenados por tamaños, de abajo hacia arriba de mayor a menor radio, esto es, con el disco más grande en la base y el más pequeño en la cima, el objetivo de juego es pasar todos los discos al tercer palo de forma que queden en el mismo orden que estaban en el primero, ordenados de abajo hacia arriba de mayor a menor radio, pero para hacerlo hay que seguir las 3 siguientes reglas: a) En cada acción sólo se puede mover un solo disco. b) Cada disco sólo puede ponerse sobre otro de mayor radio, esto es, sobre otro disco más grande, de esta manera las torres siempre tienen forma piramidal. c) Sólo pueden moverse los discos que estén en la cima del palo en el que estén insertados.

En las Torres de Hanoi a) el primer palo es la torre de partida, b) el último palo es la torre de destino y c) el palo de en medio puede considerarse de intercambio entre la torre de partida y la torre de destino. En el caso general el número de discos puede ser cualquiera, 1 ó 2, que son de solución trivial y 3, 4, 5, 6, 7, 8, ..., N, siendo 8 el número más clásico. Dado un algoritmo de resolución determinado, por ejemplo, el que utiliza TowerHanoi.Recursive o cualquier otro recursivo, el tiempo de resolución crece de forma exponencial con el número de discos. El número de movimientos mínimo a realizar para resolver el problema de la forma recursiva, que es en la que se ha programado TowerHanoi.Recursive, es de 2 elevado al número de discos menos 1. Esto es, para el caso de, por ejemplo, 5 discos es de, (2^5)-1, 31 pasos o 32 estados contando con el estado inicial de partida. Este programa TowerHanoi.Recursive incluye un función de medida de tiempos llamada HnoTime() que permite evaluar este comportamiento exponencial y que retorna una tabla de numero de discos, desde un valor inicial a uno final, y el tiempo en segundos que se ha sido necesario emplear en cada caso de resolución. Este cálculo de tiempos que realiza HnoTime() se hace

sin visualizar por pantalla los movimientos para cronometrar, de esta forma, sólo el tiempo de proceso puro.

Árbol de ficheros

TowerHanoi.Recursive soluciona de forma recursiva el problema de las Torres de Hanoi

- ← make.tol proceso de resolucin para 3 torres y cualquier numero de discos
- → 3discos.txt traza de solución por pasos para 3 discos en las torres de Hanoi
- → 4discos.txt traza de solución por pasos para 4 discos en las torres de Hanoi
- → 5discos.txt traza de solución por pasos para 5 discos en las torres de Hanoi
- → 6discos.txt traza de solución por pasos para 6 discos en las torres de Hanoi
- → 7discos.txt traza de solución por pasos para 7 discos en las torres de Hanoi
- → 8discos.txt traza de solución por pasos para 8 discos en las torres de Hanoi
- → tiempopornumerodediscos.gif gráfico del tiempo exponencial de ejecución de 1 a 22 discos
- → towerhanoi_recursive.pdf documento Pdf con el codigo de solución de las Torres de Hanoi

Declaraciones

Constantes

- `Real HnoA`
  Identificador de la torre A inicial.
- `Real HnoB`
  Identificador de la torre B intermedia o temporal.
- `Real HnoC`
  Identificador de la torre C final.
- `Real HnoN`
  Identificador del campo de numero de discos.

Funciones

- `Set StaPush(Set sta, Real ele)`
  Retorna el resultado de introducir un elemento en la cima de una pila (se considera cima, el final de un conjunto). Solo se pueden introducir en la pila elementos superiores a cero.
- `Set StaPop(Set sta)`
  Retorna un par formado por una pila y un elemento resultado de extraer el elemento de la cima de la pila sta (se considera cima, el final de un conjunto). Si se intenta sacar algo de una pila vacia, para evitar el error se devuelve una pila vacia y un cero.
- `Real StaGet(Set sta, Real pos)`
  Retorna el elemento que ocupa la posicion pos de una pila sta, si la pila tiene menos de pos elementos retorna cero.

- `Text DskText(Real max, Real rad)`
  Retorna un texto que representa un disco de ancho rad teniendo en cuenta que el ancho del disco mas ancho es max.
- `Set HnoNew(Real num)`
  Retorna un estado inicial para las torres de Hanoi con un numero num de discos, poniendolos todos en la primera torre y los mas anchos abajo.
- `Real HnoPrint(Set hno)`
  Pinta unas torres de Hanoi y retorna el numero de discos pintados.
- `Set HnoMove(Set hno, Real from, Real to, Real trz)`
  Retorna unas nuevas torres de Hanoi resultado de mover el disco de la cima de la torre from a la torre to.
- `Set HnoSolve(Set hno, Real trz, Real tim)`
  Resuelve las torres de Hanoi y retorna un conjunto formado por el estado final y el tiempo empleado. Este metodo de resolucion funciona cuando las torres se encuentran en el estado inicial, todas en la primera torre.
- `Set HnoTime(Real ini, Real end)`
  Retorna una tabla de tiempos resultado de resolver el problema de las torres de Hanoi desde ini torres hasta end torres.

Pruebas

- `Set HnoT03`
  Resuelve el caso de 3 discos.
- `Set HnoT04`
  Resuelve el caso de 4 discos.
- `Set HnoT05`
  Resuelve el caso de 5 discos.
- `Set HnoT06`
  Resuelve el caso de 6 discos.
- `Set HnoT07`
  Resuelve el caso de 7 discos.
- `Set HnoT08`
  Resuelve el caso de 8 discos.
- `Set HnoTim`
  Mide tiempos de ejecucion de 1 a 20 discos.

Constantes

Real HnoA

```
////////////////////////////////////////////////////////////////////////////
Real HnoA = 1;
////////////////////////////////////////////////////////////////////////////
PutDescription("Identificador de la torre A inicial.",HnoA);
////////////////////////////////////////////////////////////////////////////
```

Real HnoB

```
////////////////////////////////////////////////////////////////////////////
Real HnoB = 2;
////////////////////////////////////////////////////////////////////////////
PutDescription("Identificador de la torre B intermedia o temporal.",HnoB);
////////////////////////////////////////////////////////////////////////////
```

Real HnoC

```
////////////////////////////////////////////////////////////////////////////
Real HnoC = 3;
////////////////////////////////////////////////////////////////////////////
PutDescription("Identificador de la torre C final.",HnoC);
////////////////////////////////////////////////////////////////////////////
```

Real HnoN

```
////////////////////////////////////////////////////////////////////////////
Real HnoN = 4;
////////////////////////////////////////////////////////////////////////////
PutDescription("Identificador del campo de numero de discos.",HnoN);
////////////////////////////////////////////////////////////////////////////
```

Set StaPush()

```
////////////////////////////////////////////////////////////////////////////
Set StaPush(Set  sta, // Pila (stack)
            Real ele) // Elemento
////////////////////////////////////////////////////////////////////////////
{ If(ele, sta << [[ ele ]], sta) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el resultado de introducir un elemento en la cima de una pila (se
considera cima, el final de un conjunto).
Solo se pueden introducir en la pila elementos superiores a cero.",
StaPush);
////////////////////////////////////////////////////////////////////////////
```

Set StaPop()

```
////////////////////////////////////////////////////////////////////////////
Set StaPop(Set  sta) // Pila (stack)
////////////////////////////////////////////////////////////////////////////
{
  Real car = Card(sta);                                  // Numero de elementos
  If(Not(car), [[ Empty, 0 ]],
  {
    Set  res = For(1, car-1, Real(Real pos){ sta[pos] }); // Resto de la pila
    Real cim = sta[car];                                  // Cima de la pila
    [[ res, cim ]]
  })
};
////////////////////////////////////////////////////////////////////////////
```

```
PutDescription(
"Retorna un par formado por una pila y un elemento resultado de extraer el
elemento de la cima de la pila sta (se considera cima, el final de un
conjunto).
Si se intenta sacar algo de una pila vacia, para evitar el error se devuelve
una pila vacia y un cero.",
StaPop);
////////////////////////////////////////////////////////////////////////////
```

Real StaGet()

```
////////////////////////////////////////////////////////////////////////////
Real StaGet(Set  sta, // Pila (stack)
            Real pos) // Posicion
////////////////////////////////////////////////////////////////////////////
{ If(GT(pos,Card(sta)), 0, sta[pos]) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el elemento que ocupa la posicion pos de una pila sta, si la pila
tiene menos de pos elementos retorna cero.",
StaGet);
////////////////////////////////////////////////////////////////////////////
```

Text DskText()

```
////////////////////////////////////////////////////////////////////////////
Text DskText(Real max, // Ancho maximo
             Real rad) // Radio del disco
////////////////////////////////////////////////////////////////////////////
{
  Text space = Repeat(" ",max-rad);
  Text disk  = Repeat("=",rad);
  space+disk+"|"+disk+space
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un texto que representa un disco de ancho rad teniendo en cuenta
que el ancho del disco mas ancho es max.",
DskText);
////////////////////////////////////////////////////////////////////////////
```

Set HnoNew()

```
////////////////////////////////////////////////////////////////////////////
Set HnoNew(Real num) // Numero de discos
////////////////////////////////////////////////////////////////////////////
{ [[ Range(num, 1, -1), Empty, Empty, num ]] };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un estado inicial para las torres de Hanoi con un numero num de
discos, poniendolos todos en la primera torre y los mas anchos abajo.",
HnoNew);
////////////////////////////////////////////////////////////////////////////
```

Real HnoPrint()

```
////////////////////////////////////////////////////////////////////////////
Real HnoPrint(Set hno) // Torres de Hanoi
////////////////////////////////////////////////////////////////////////////
{
  Text WriteLn("Hanoi:");
  Set ran = Range(hno[HnoN], 1, -1);
```

```
  Set wri = EvalSet(ran, Real(Real pos)
  {
    Text Write  (" "+DskText(hno[HnoN],StaGet(hno[HnoA],pos)));
    Text Write  (" "+DskText(hno[HnoN],StaGet(hno[HnoB],pos)));
    Text WriteLn(" "+DskText(hno[HnoN],StaGet(hno[HnoC],pos)));
    pos
  });
  Card(wri)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Pinta unas torres de Hanoi y retorna el numero de discos pintados.",
HnoPrint);
////////////////////////////////////////////////////////////////////////////
```

Set HnoMove()

```
////////////////////////////////////////////////////////////////////////////
Set HnoMove(Set  hno,  // Torres de Hanoi
            Real from, // Posicion de la que se saca un disco
            Real to,   // Posicion a la que se lleva un disco
            Real trz)  // Si TRUE pinta el resultado tras el movimiento
////////////////////////////////////////////////////////////////////////////
{
  Set tup = StaPop(hno[from]);       // para resto de pila y elemento
  Set res = tup[1];                  // Torre con uno menos
  Set add = StaPush(hno[to],tup[2]); // Torre con un elemento mas

  Set for = For(1, 3, Set(Real pos)  // Reconstruir la nueva estructura
  {
    If(EQ(pos,from), res,
    If(EQ(pos,to),   add,
                     hno[pos]))
  });
  Set  new = for << [[ hno[HnoN] ]];
  Real wri = If(trz, HnoPrint(new), FALSE);
  new
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna unas nuevas torres de Hanoi resultado de mover el disco de la cima
de la torre from a la torre to.",
HnoMove);
////////////////////////////////////////////////////////////////////////////
```

Set HnoSolve()

```
////////////////////////////////////////////////////////////////////////////
Set HnoSolve(Set  hno, // Torres de Hanoi
             Real trz, // Si TRUE visualiza una traza mientras resuelve
             Real tim) // Si TRUE toma tiempos
////////////////////////////////////////////////////////////////////////////
{
  Set solve(Set hno, Real from, Real to, Real tmp, Real dsk)
  {
    Real If(Not(trz), FALSE,
    { // Traza la recursion
      Text WriteLn("Disk "+FormatReal(dsk, "%.0lf")+": "+
                           FormatReal(from,"%.0lf")+"->"+
                           FormatReal(to,  "%.0lf"));
       TRUE
    });

    If(EQ(dsk, 1), HnoMove(hno, from, to, trz),
    {
      Set step01 = solve(hno, from, tmp, to, dsk-1);
      Set step02 = HnoMove(step01, from, to, trz);
```

```
      Set step03 = solve(step02, tmp, to, from, dsk-1);
      step03
    })
  };

  Real If(trz, HnoPrint(hno), FALSE);  // Pinta el estado inicial
  Real ini = Copy(Time);

  Set  res = solve(hno, 1, 3, 2, hno[HnoN]); // Resuelve

  Real sec = Copy(Time)-ini;                 // Tiempo empleado
  Real If(Not(tim), FALSE,
  {
    WriteLn("Torres "+FormatReal(hno[HnoN],"%3.0lf")+
            ": "     +FormatReal(sec,      "%3.0lf")+" secs");
    TRUE
  });

  [[ res, sec ]]
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Resuelve las torres de Hanoi y retorna un conjunto formado por el estado
final y el tiempo empleado.
Este metodo de resolucion funciona cuando las torres se encuentran en
el estado inicial, todas en la primera torre.",
HnoSolve);
////////////////////////////////////////////////////////////////////////////
```

Set HnoTime()

```
////////////////////////////////////////////////////////////////////////////
Set HnoTime(Real ini, // Numero de torres iniciales
            Real end) // Numero de torres finales
////////////////////////////////////////////////////////////////////////////
{
  // Calcular los tiempos
  Set for = For(ini, end, Set(Real numDsk)
            { HnoSolve(HnoNew(numDsk), FALSE, TRUE) });

  // Obtener numero de torres y resultados
  Set dat = EvalSet(for, Set(Set res) { [[ res[1][HnoN], res[2] ]] });

  // Poner una cabecera y visualizar el grafico
  Set heaDat = [[ [[ "Hanoi", "towers, seconds" ]] ]] << dat;

  // Podria visualizarse el grafico
  Set chr = Chart(heaDat, "");

  heaDat
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna una tabla de tiempos resultado de resolver el problema de las torres
de Hanoi desde ini torres hasta end torres.",
HnoTime);
////////////////////////////////////////////////////////////////////////////
```

Set HnoT03

```
////////////////////////////////////////////////////////////////////////////
Set HnoT03 = If(FALSE, HnoSolve(HnoNew(3),  TRUE,  TRUE), Empty);
////////////////////////////////////////////////////////////////////////////
PutDescription("Resuelve el caso de 3 discos.", HnoT03);
////////////////////////////////////////////////////////////////////////////
```

Set HnoT04

```
////////////////////////////////////////////////////////////////////////////
Set HnoT04 = If(FALSE, HnoSolve(HnoNew(4),  TRUE,  TRUE), Empty);
////////////////////////////////////////////////////////////////////////////
PutDescription("Resuelve el caso de 4 discos.", HnoT04);
////////////////////////////////////////////////////////////////////////////
```

Set HnoT05

```
////////////////////////////////////////////////////////////////////////////
Set HnoT05 = If(FALSE, HnoSolve(HnoNew(5),  TRUE,  TRUE), Empty);
////////////////////////////////////////////////////////////////////////////
PutDescription("Resuelve el caso de 5 discos.", HnoT05);
////////////////////////////////////////////////////////////////////////////
```

Set HnoT06

```
////////////////////////////////////////////////////////////////////////////
Set HnoT06 = If(FALSE, HnoSolve(HnoNew(6),  TRUE,  TRUE), Empty);
////////////////////////////////////////////////////////////////////////////
PutDescription("Resuelve el caso de 6 discos.", HnoT06);
////////////////////////////////////////////////////////////////////////////
```

Set HnoT07

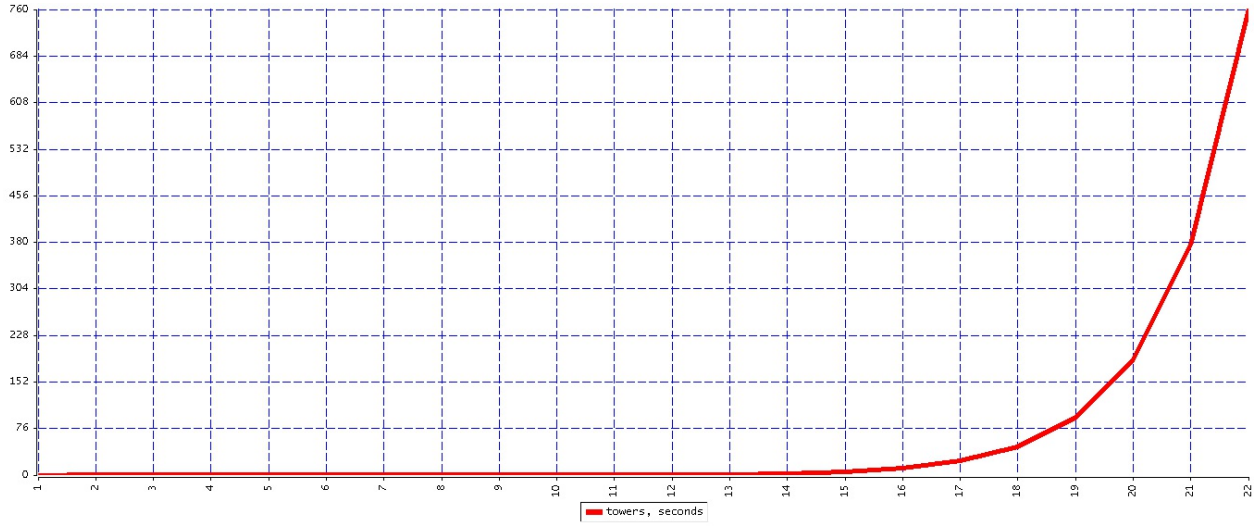
```
////////////////////////////////////////////////////////////////////////////
Set HnoT07 = If(FALSE, HnoSolve(HnoNew(7),  TRUE,  TRUE), Empty);
////////////////////////////////////////////////////////////////////////////
PutDescription("Resuelve el caso de 7 discos.", HnoT07);
////////////////////////////////////////////////////////////////////////////
```

Set HnoT08

```
////////////////////////////////////////////////////////////////////////////
Set HnoT08 = If(FALSE, HnoSolve(HnoNew(8),  TRUE,  TRUE), Empty);
////////////////////////////////////////////////////////////////////////////
PutDescription("Resuelve el caso de 8 discos.", HnoT08);
////////////////////////////////////////////////////////////////////////////
```

Set HnoTim

```
////////////////////////////////////////////////////////////////////////////
Set HnoTim = If(TRUE, HnoTime(1,22), Empty);
////////////////////////////////////////////////////////////////////////////
PutDescription("Mide tiempos de ejecucion de 1 a 20 discos.", HnoTim);
////////////////////////////////////////////////////////////////////////////
```


760
684
608
532
456
380
304
228
152
76
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
towers, seconds