Ventajas de la Programación Funcional en el Aprendizaje y Enseñanza de las Matemáticas

Pablo López y Blas Ruiz
Dpto. Lenguajes y Ciencias de la Computación
Universidad de Málaga

¿ Qué entendemos por matemática ?

Matemática números ecuaciones y funciones

$\mathbb{N}$ $\mathbb{Z}$ $\mathbb{Q}$ $\mathbb{R}$ $x^3+y^3=z^3$

Lógica Teoría de Números T. de Grupos
Geometría Análisis Matemático ...

Lenguajes de Programación

Java Python Ruby C++ Haskell Prolog

Computación

Inteligencia Artificial Lenguajes Formales
Compiladores Criptografía y Ciberseguridad

La descripción de una función captura los cómputos

Aprender programando con funciones

programar = escribir funciones

los números y las funciones son la esencia de la Matemática

Matemática (definición de las funciones)

Programación con funciones

Computación (evaluar las funciones)

* El lenguaje matemático es universal
* En nuestro lenguaje funcional la descripción de las funciones es "similar"

La descripción de una función captura los cómputos

¿ un Lenguaje Funcional ?

¿ Qué ventajas aporta ?

SON CERCANOS A LA MATEMÁTICA (notación, corrección y disciplina)

¿ Haskell ? (en honor a Haskell Curry)

Tras Hugs y Miranda (1987-90), es el lenguaje funcional que enseñamos a Ingenieros en Informática, Matemáticos y Doble Grado II-MAT desde 1990.

- funciones con tipos genéricos completamente seguras
- estructuras "infinitas" y evaluación PEREZOSA (criba de Eratóstenes)

Qué es la matemática

μάθημα (aprender vía instrucción)

Representar objetos y las relaciones entre estos, a través de expresiones y funciones (esos objetos pueden ser a su vez funciones)

¿Y la programación funcional?

Describir cómputos con expresiones y funciones

Ejemplo introductorio:

* calcular el número de filas de la menor caja de bombones piramidal que permite alojar 132 bombones
* calcular la altura de un racimo de uvas piramidal

Las sucesiones son esenciales

Matemática discreta (ecuaciones diofánticas, combinatoria ...)
Análisis Matemático, Teoría de juegos, Computación/Programación ...

Una sucesión es una función $s : \mathbb{N} \rightarrow A$

$$[s(0)\,,\ s(1),\ \ s(2)\ldots\ ]$$

Para definirla usaremos una ley o fórmula:

1. – $\sigma(n) = (-1)^n$
2. – $f(n) = n\,f(n-1),\ n > 0;\qquad f(0) = 1 \qquad factorial$
3. – $c(n) = n^2 + c(n-1),\ n > 0;\ c(0) = 0 \qquad suma\ de\ cuadrados$

2 y 3 son definiciones recurrentes

Un ejemplo: factorial

PROBLEMA: ¿ De cuántas formas pueden salir en fila 59 escolares ? ... 59!

si sé calcular 58!, entonces 59! = 59 * 58!
y si sé calcular 57!, entonces 58! = 58 * 57!, y si ...

Nuestra "super-calculadora" Haskell

```
Prelude> 7*6*5*4*3*2*1
5040
Prelude> [7,6..1]       -- secuencia aritmética
[7,6,5,4,3,2,1]
Prelude> product [7,6..1]        →   7 * product [6,5..1]  →  7 * 6 * product [5,4 .. 1]  → ...
5040
Prelude> product [59,58..1]              -- 59!
13868311854568983573793901972038940634590287677268743254082129494016000000
0000000                          ¿Es un número muy grande?
Prelude> 10^80
10000000000000000000000000000000000000000000000000000000000000000000000000
0000000
Prelude> product [59,58..1] > 10^80     ← NÚMERO DE ÁTOMOS DEL UNIVERSO
True
```

Un ejemplo: factorial (2)

¿Cuántas cifras tiene el factorial?

```
Prelude> product [59,58..1]
13868311854568983573793901972038940634590287677268743254082129494016000000
0000000

Prelude> length $ show $ product [59,58..1]
81
```

También podemos usar logaritmos para calcular el número de cifras:

recuerda: b^q = N <==> logBase b N = q : el exponente al que hay que elevar ...

```
Prelude> logBase 10 (product [59,58..1])      -- logBase 10 determina el número de dígitos en
base 10
80.14202359900654              10^80 <=  59!  < 10^81
```

¿ con cuántos ceros termina 59! ?

Un ejemplo: factorial (3)

Sin funciones auxiliares podemos definir el factorial ☺

(lenguaje matemáticas)

$_ ! : Z \rightarrow Z$

n ! = 0 , n=0

= n (n-1)! , n > 0

(Haskell)

```
factorial ::  Integer -> Integer
factorial n | n==0  = 1
            | n > 0  = n * factorial (n-1)
```

"n" es una variable, en matemáticas y en Haskell

recursión : en su definición aparece la función a definir (recurrencia)

definición con guardas : | n> 0 = “expresión_protegida”

☛ sin miedo

factorial 3 =

3 * factorial 2 = en esta "llamada" n denota el valor 2

3 * (2 * factorial 1) =

3 * (2 * (1 * factorial 0)) =

3 * (2 * (1 * 1)) = !! y ahora hacia atrás

3 * (2 * 1) = ...

Un ejemplo: factorial (4)

MUY IMPORTANTE

CORRECCIÓN: la definición es correcta:

una llamada termina siempre, y calculando n!

```
*Charla> factorial 1000 > 1        -- el operador (>) obliga a calcular 1000!
True
it :: Bool                         TIPO DEL RESULTADO
(0.00 secs, 963,008 bytes)         TIEMPO y MEMORIA USADA
```

PROBLEMA 1 : ¿ cuántos ceros finales tiene 1000! ?

UNA ESTRATEGIA "A BOTE PRONTO": calcular, y contar

```
*Charla> (show . factorial) 42                -- composición de funciones
"1405006117752879898543142606244511569936384000000000"

*Charla> (reverse . show . factorial) 42
"000000000483639965115442606241345898978257711600 5041"

*Charla> (takeWhile (=='0') . reverse . show . factorial) 42
"000000000"

*Charla> (length . takeWhile (=='0') . reverse . show . factorial) 42
9
```

Así pues definimos la función:

```
cerosFactorial  = length . takeWhile (=='0') . reverse . show . factorial
```

problema resuelto, pero ... ¿ es una solución EFICIENTE ?

PROBLEMA 1 : ¿ cuántos ceros finales tiene 1000! ? (2)

La mejor estrategia: contar cuántos cincos aparecen en el producto sin evaluarlo:

1 ... 5 ... 2*5 ... 3*5 .... 4*5 ... 5^2 ... 6*5 ... . 9*5 ... 5^3 ..... n-1 n

¿Cuántos cincos aparecen?

uno de cada 5:	n `div` 5
otro adicional por cada 5^2:	n `div` (5^2)
otro adicional por cada 5^3:	n `div` (5^3)
...	

Luego, hay que evaluar la suma :

n `div` 5 + n `div` (5^2) + n `div` (5^3) + ...

y dejamos de sumar al encontrar el primer sumando nulo

Ejemplo: 137! 137 = 27*5 + 2 137 = 5*25 + 12 137 = 1*125 + 12 , 137 = 0*625 + 137

137 `div` 5 + 137 `div` (5^2) + 137 `div` (5^3) = 27 + 5 + 1 = 33 ceros !!

```
cerosFactorial = sum . takeWhile (/= 0) . cincos

cincos n = [ n `div` (5^k) | k <- [1,2 ..] ]
                -- observa:  [  n `div` 5 ,  n `div` (5^2) ,  n `div` (5^3)  ... , 0, 0 ...  ]
```

PROBLEMA 1 : ¿ cuántos ceros finales tiene 1000! ? (3)

Podemos evaluar la suma

n `div` 5 + n `div` (5^2) + n `div` (5^3) + ...

por exceso, tomando la expresión:

n/5 + + n / 5^2 + n / 5^3 + ... =

n(1/5 + 1/25 +1/ 125 + ...) =

nS, donde S = 1/5 + 1/25 +1/ 125 + ...

y de aquí 5S = 1 + (1/5 + 1/25 + ...) = 1+S

y finalmente 5S=1+S => S = 1/4 , y de aquí

n/5 + + n / 5^2 + n / 5^3 + ... = n/4

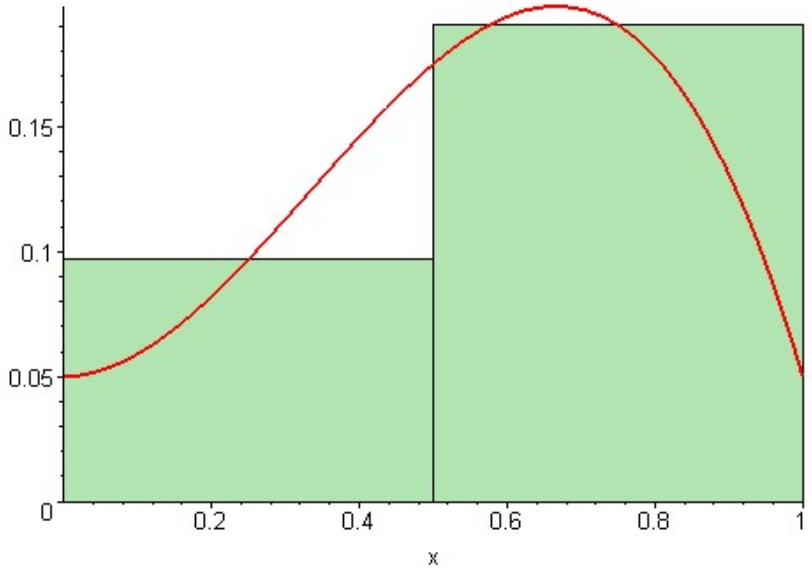
Luego el valor n/4 es una cota superior del número de ceros finales de n!, que resulta ser bastante ajustada:

*Charla> [(n,cerosFactorial'' n, fromIntegral n/4) | n <- [10,20..100]]
[(10,2,2.5),(20,4,5.0),(30,7,7.5),(40,9,10.0),(50,12,12.5),(60,14,15.0),(70,16,17.5),(80,19,20.0),(90,21,22.5),(100,24,25.0)]

Problema 2: área bajo una curva (integral)

Dada y=f(x), calcular el área correspondiente al intervalo [a,b] , con cierta precisión

ESTRATEGIA: dividimos el intervalo en dos partes, calculamos el área de cada trozo y sumamos;



A su vez, cada parte puede evaluarse con la misma estrategia en forma iterativa o recursiva.

Problema 2: área bajo una curva (integral)

ESTRATEGIA: dividimos el intervalo en dos partes,
calculamos el área de cada trozo y sumamos

```
área a b ep f          --  a y b son los extremos del intervalo,  ep es la precisión

  | abs dif  >  ep     = área a c ep f    +    área c b ep f

  | otherwise         = dif * (f c)   -- si el intervalo es "pequeño" devolvemos el área
                                       -- del rectángulo
 where  dif = b-a           -- ancho del rectángulo
        c   = (a+b)/2       -- punto medio del intervalo
```

```
-- área limitada por la curva y = x^2, el eje OX y las rectas x=-1, x=4
*Charla> área (-1) 4  0.001 (^2)
21.666666511446238

-- si recordamos la regla de Barrow del cálculo integral,
*Charla> (  (^3) 4 -  (^3) (-1)  ) / 3.0
21.666666666666668
```

Qué es la matemática (2)

La lección de Carl Gauss (1777-1855)

$n \mapsto s(n)$
…
$4 \mapsto 10$
$3 \mapsto 6$
$2 \mapsto 3$
$1 \mapsto 1$

1	2	3	….	99	100
100	99	98	…	2	1
101	101	101		101	101

μάθημα (aprender)

$$\frac{101(100)}{2} = 5050 \qquad s(n) = \frac{n(n+1)}{2}$$

"calcular el número de filas de la menor caja de bombones piramidal con 132 bombones (o el menor racimo)"

Sea t = nº de uvas/bombones; aproximamos: $\frac{x(x+1)}{2} = t$

$$x(x+1) - 2t = 0;\ x^2 + x - 2t = 0;\quad x = \frac{-1+\sqrt{1+8t}}{2}$$

Para t = 24 , x = 6.446221994724902 $n = \left\lceil \frac{-1+\sqrt{1+8t}}{2} \right\rceil$ ☺ el mejor programa

Problema 3: Racimos de uvas piramidales

Racimo : todos los pisos completos salvo el último

1 + 2 + 3 + 4 + 3 = 13

Cuestiones:

(A) ¿Cuántas uvas como máximo puede contener un racimo de altura h ?

(B) Conocido el número de uvas t, calcular la altura del racimo

(A) El máximo de es una progresión aritmética: t = h*(h+1) `div` 2

(B) Conocido t > 0, aproximamos h y vía la solución cde una ecuación de 2º grado:

h = (-1 + sqrt (1+8*t))/ 2, por ejemplo si h=24:

```
*Charla>  ( -1 + sqrt ( 1+8*24) )  / 2
6.446221994724902                  --    6 < h  <=  7
```

```
altura t  = ceiling $ ( -1 + sqrt ( 1+8* fromInteger t) ) / 2
```

```
*Charla> altura 28
 7                  1 + 2 + 3 + 4 + 5 + 6 + 7
*Charla> altura 29
8                   1 + 2 + 3 + 4 + 5 + 6 + 7 + 1
```

HEMOS ESCRITO EL MEJOR PROGRAMA ☺

Problema 5: Números primos

Un número natural se dice que es primo si tiene exactamente dos divisores naturales distintos; e.d., tiene un solo divisor propio, el 1 [1 no es primo]

Euclides: Elementos (177 a. C., Alejandría)

El conjunto de números primos es infinito (proposición 20, libro IX, Elementos)

Estos números son fundamentales en Matemáticas y Computación:

Teorema fundamental de la aritmética

Todo número natural se descompone en productos de números primos (Euclides, proposición 4, libro 9)

y la descomposición es única salvo el orden de los factores [1 no es primo]
(Carl Gauss, 1801, Disquisitiones Arithmeticae)

Problema 5: Números primos (2)

```
esPrimo :: Integer -> Bool
esPrimo n  =   [1] == divisoresPropios n          -- 1 es el único divisor propio

divisoresPropios n = [ d |  d <- [1 .. n `div` 2],   n `mod` d == 0 ]
-- si n = d d'   ,  y  d > 1 es divisor  propio, no puede superar a  la mitad de n

primos = 2 : filter esPrimo [3,5.. ]        -- lista infinita de los números primos

*Charla> take 20 primos
[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71]
```

Problema 5: Números primos (3)

La criba de Eratóstenes

Eratóstenes (Cirene, 276 - Alejandría, 194 a. C.) , matemático y astrónomo (calculó: distancia al Sol, inclinación y diámetro de la Tierra, y propuso el año bisiesto).

Descubrió la Criba de Eratóstenes, método estándar usado para comparar diferentes lenguajes de programación. Pero ...

¿y si queremos la lista infinita de primos?

```
criba []     =  []
criba (x:xs) =  x : criba ( quitaMúltiplos x xs)

quitaMúltiplos x xs = [ y |y <- xs , y `mod` x /=0 ]
primosCriba = criba  [2.. ]

*Charla> length $ take 5000 $ primosCriba
5000
(5.93 secs, 4,410,814,288 bytes)                6' -    4.5 Gb
*Charla> length $ take 5000 $ primos
5000
(31.13 secs, 15,671,523,728 bytes)              31'  - 15.5 Gb
```

Problema 5: Números primos (4)

Números de Euclides. El conjunto de primos es infinito

Si p_n es la sucesión de números primos, el n-ésimo número de Euclides es:

1 + p_1 p_2 ... p_n

e n = 1 + product (take n primos)

e 1 -> 1+2 = 3 e 2 -> 1 + 2*3 = 7 e 3 -> 31 ...

Los primeros euclidianos son : 3, 7, 31, 211, 2311, 30031, 510511 ...

Muchos son primos, pero 30031 = 59*509 no es primo ¿ y el siguiente primo?

```
numerosDeEuclidesPrimos = [  (n , f, esPrimo f) | n <- [1..1000], let f = e n ]

-- *Charla> numerosDeEuclidesPrimos
--[(1,3,True),(2,7,True),(3,31,True),(4,211,True),(5,2311,True),(6,30031,False),(7,510511,False)
--,(8,9699691,False),(9,223092871,False),(10,6469693231,False),(11,200560490131,True?
```

CONJETURAS: hay infinitos números de Euclides primos.
ningún número de Euclides es un cuadrado perfecto.

Problema 5: Números primos (5)

Un número es **biprimo** si es primo y su invertido es también primo

```
valorInvertido = read . reverse . show
*Charla> valorInvertido 1234
4321

esBiprimo n =  esPrimo n && esPrimo (valorInvertido n)

*Charla> filter esBiprimo [99000 .. 100000]
[99023,99053,99109,99119, ...  ,99431,99563,99571,99611,99661,99713,99721,99793,9981
7,99829,99877,99881,99907,99923,99989]
```

La conjetura de GoldBach

Con Haskell podemos construir programas inmediatos para analizar conjeturas:

La conjetura de Christian Goldbach (¿ Descartes, 163X ?) :

"Todo número par mayor que 2 es suma de dos números primos."

Christian Goldbach (en una carta a Euler, en 1742)

```
goldbach  n = not $ null [ p |  p <- [2..n `div` 2], esPrimo p, esPrimo (n-p) ]

conjeturaGoldbach  a b =  and [goldbach n | n <- [ a, a+2 .. b] ]
-- *Charla> conjeturaGoldbach 4 5000
-- True                                          -- verificada hasta  4*10^18 (2018)
```

Conjetura de Wouf Gaia (2014)

"Todo número par mayor que 4 es suma de un primo y un biprimo"

```
gaia  n = not $  null [ (p,n-p) |  p <- [2..n-2], esBiprimo p, esPrimo (n-p) ]

conjeturaGaia a b =  and [gaia n | n <- [ a, a+2 .. b]    ]
-- *Charla> conjeturaGaia 10 10000
-- True
```

La conjetura de GoldBach (2)

MATEMÁTICAS RECREATIVAS Y PROGRAMACIÓN RECREATIVA

Un árbitro elige dos números distintos entre 2 y 99, calcula el producto y la suma, los escribe en dos papeles distintos, entrega el papel con el producto a una persona PROD y el otro a SUM; después de leer y analizar los papeles, mantienen el siguiente diálogo:

PROD.— No puedo adivinar tu suma.
SUM.— Ya lo sabía; yo tampoco puedo adivinar tu producto.

PROD.— ¡Ajá!, entonces ya sé tu suma.
SUM.— En ese caso, yo también conozco tu producto.

¿Cuáles fueron los números elegidos por el árbitro?

1.- PROD no tiene el producto de dos primos.
2.- **Según la conjetura de Goldbach**, SUM tampoco tiene una suma par !!

El problema P-S de McCarthy y otros acertijos; B. Ruiz, F. Gutiérrez, J. Gallardo; Suma 21; Febrero 1999 (pp. 21-33)
https://redined.educacion.gob.es/xmlui/handle/11162/12910?show=full

Problema 5: Números primos (6)

Números perfectos

perfecto n = sum (divPropios n)) == n -- ejemplo 1+2+3 = 6; 6 es perfecto

-- *Charla> filter perfecto [1..] -- todos los números perfectos
-- [1,6,28,496,8128, -- el siguiente es 3 3550 336

Números primos de Mersenne $2^p - 1$

Marin Mersenne: matemático, filósofo y sacerdote francés del S. XVII

Euclides-Euler: si $2^p - 1$ es primo, entonces $2^{p-1}(2^p - 1)$ es perfecto
los únicos perfectos pares son los anteriores (primos de Mersenne)

Conjeturas de Euler: ningún impar es perfecto existen infinitos perfectos

primosMer = [(n,m,ep) | n <- primos , let m = 2^n - 1, let ep = esPrimo m]
*Charla> primosMer
[(2,3,True),(3,7,True),...(11,2047,False), (23,8388607,False), ...(31,2147483647 [Euler] ...
El mayor conocido (2018) es 2^82589933 - 1, y tiene 24.862.048 cifras

Problema 6: Subir la escalera

Propuesto en la Olimpíada Asturias 2022:
https://impulsotic.org/xii-olimpiada-de-ingenieria-informatica-asturias-2022-pruebas/

A Pedro le gusta subir las escaleras saltando 3 o 5 escalones a la vez.
(a) ¿ Puede subir una escalera de 16 peldaños ?
(b) ¿ De cuántas maneras distintas puede hacerlo ?
¿ (a) es más sencillo que (b) ?

Estrategia para contar:

```
escalera n     -- devuelve las formas distintas de subir una escalera de n peldaños:
  | n==0   = 1
  | n>0    = escalera (n-3) + escalera (n-5)
  | n<0    = 0
```

```
*Charla> escalera 11
3     ::  corresponde a las 3 secuencias distintas de saltos  [3,3,5] ,  [3,5,3]  ,  [5,3,3]
```

Problema 6: Subir la escalera (2)

```
puedoSubirLaEscalera n  = escalera n > 0
-- muy ineficiente:  hay que evaluar escalera n, luego todas las sumas

    UNA MEJORA ESPECTACULAR !!  :   dejar de "explorar" al encontrar una solución

puedoSubirLaEscalera 0  = True      -- ya está arriba
puedoSubirLaEscalera n              -- (en saltos de  3 y 5)
   | n>0                    =   puedoSubirLaEscalera (n-3) ||  puedoSubirLaEscalera (n-5)
   | n<0                    =  False

True  || b = True      --    ||  ES ESTRICTO en el primer argumento
False || b = b

escalerasQuePuedoSubir =  filter puedoSubirLaEscalera [ 1 .. ]

*Charla> take 100 $ escalerasQuePuedoSubir
[3,5,6,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23...
    -- ¿ podré subir cualquier escalera con más de 7 escalones ?
```

Para responder NECESITAMOS... ?

Problema 6: Subir la escalera (3)

*Charla> take 100 $ escalerasQuePuedoSubir
[3,5,6,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23...
-- puedo subir cualquier escalera de más de 7 escalones **¿por qué?**

Debemos estudiar la ecuación diofántica para $n \in \mathbb{N}$:
[Diofanto de Alejandría, S III]

$$\exists \alpha, \beta \in \mathbb{N} : \; n = 3\alpha + 5\beta \qquad \heartsuit$$

n verifica ♥ si y solo si es posible subir la escalera de n peldaños

Demostremos por inducción que la ecuación tiene solución para todo natural $n \geq 8$:

1.- Si $n = 8 = 3(1) + 5(1)$, luego n=8 es solución
2.- Si $n \geq 8$ verifica ♥, entonces $n + 1$ verifica ♥

$$\begin{array}{l} n = 3\alpha + 5\beta, \quad n \geq 8 \\ \underline{1 = 3(2) + 5(-1)} \\ n+1 = 3(\alpha+2) + 5(\beta-1) \\ \underline{0 = 3(-5) + 5(3)} \\ n+1 = 3(\alpha-3) + 5(\beta+2) \end{array}$$

Si $\beta = 0$ y $n = 3\alpha \geq 8$, debe tenerse $\alpha \geq 3$
¡ sumamos 0 !
luego la representación es válida ☺

Problema 8: Crecimiento de una población

Leonardo de Pisa (1170 - 1240), Fibonacci (filius Bonacci), en su texto Liber abaci, introduce el siguiente modelo de crecimiento:

- en cada generación, una pareja de adultos tiene dos descendientes (críos)
- en la siguiente generación los críos pasan a ser adultos

Si partimos de 4 críos y 8 adultos,

¿cuántos individuos tendrá la generación 100 ?

	críos	adultos
poblacion inicial	4	8
1ª generación	8	12
2ª gen.	12	20
3ª	20	32
...		
n	c	a
n+1	a	c+a
...		

```
-- fib n |   n> 1 =  fib(n-2) + fib (n-1)
```

```
fibs c a  =   c : fibs a (c+a)        -- fibs 1  2  -> 1:fibs 2 3  ->  1:2:fibs 3 5 ...
fib a b n =   fibs a b  !! n          --  logBase 10 $ fromInteger $ fib 1 2 100     ->    20.97 ...
```

Problema 9: El mono y los cocos

Tres náufragos y un mono conviven en una isla. Pasan la tarde recolectando cocos para el día siguiente. Un náufrago se despierta de noche y tiene hambre. Divide en tres montones el total de cocos y como sobra un coco, lo entrega al mono. Se come su parte y deja la dos partes restantes en el saco. Al rato se despierta otro náufrago que actúa de igual modo, y dos horas mas tarde hará lo mismo el tercer náufrago. Por la mañana dividen los cocos del saco en tres montones justos. ¿Cuántos cocos habían recolectado?

Veamos las operaciones necesarias para comprobar si x es solución

```
cocos3 :: Integer -> Bool
cocos3    x        =  r1==1 &&  r2==1 && r3==1 && r4==0
 where
   (x1,r1) = divMod x 3                  --      x   == 3 * x1 + 1
   (x2,r2) = divMod (2*x1) 3             -- 2*x1   == 3 * x2 + 1
   (x3,r3) = divMod (2*x2) 3             -- 2*x2   == 3 * x3 + 1
   (x4,r4) = divMod (2*x3) 3             -- 2*x3   == 3 * x4
```

r1 es el primer valor que se evalúa

```
solucionesCocos3 = filter cocos3 [1..]     ESTA FUNCIÓN resuelve el problema
*Charla> solucionesCocos3
[25,106,187,268,349,430,511,592,...    -- progresión aritmética de difer. 3^4  ¡AJÁ!
*Charla> take 10  $ solucionesCocos5   -- SUSPECHA!:     y para 5 náufragos ?
[3121,18746,34371,...                  -- progresión de diferencia 5^6    ¡AJAAAAAAÁ!
```

Problema 9: El mono y los cocos (2)

Conjetura: el conjunto de soluciones es una progresión de diferencia 3^4

Solución general del sistema de ecuaciones diofánticas

$$
\begin{aligned}
x &= 3x_1 + 1 && (*\ 8) && 8x &= 24x_1 + 8\\
2x_1 &= 3x_2 + 1 && (*\ 12) && 24x_1 &= 36x_2 + 12\\
3x_2 &= 3x_3 + 1 && (*\ 18) && 36x_2 &= 54x_3 + 18\\
2x_3 &= 3x_4 && (*\ 27) && 54x_3 &= 81x_4\\
& && && 8x &= 81x_4 + 38
\end{aligned}
$$

sumamos las ecuaciones y simplificamos:

Eliminemos el "8" que acompaña a x; para ello pongamos $x_4 = 8z + r \quad 0 \le r < 8$ entonces:

$$8x = 81(8z + r) + 8*4 + 6 = \underline{8(81z + 4) + 80r} + r + 6$$

De aquí $r = 2$, luego

$x = 81z + 25$ una progresión de diferencia $81 = 3^4$

Lecturas recomendadas

Elementos. Libros I-IV; **Euclides**; Biblioteca Clásica Gredos nº 155 (1991);
trad. de María Luisa Puertas Castaños

MATEMÁTICAS RECREATIVAS Y PROGRAMACIÓN RECREATIVA

What Is the Name of This Book?; **Raymond M. Smullyan** (1978)
(¿Cómo se llama este libro? El enigma de Drácula y otros pasatiempos)
(https://webooks.co/images/team/generos/literarios495/)

Matematicas Recreativas; **Y. Perelman** (trad. Martinez Roca , 1968)
http://www.librosmaravillosos.com/matematicarecreativa/#capitulo06
(matemáticas, aritmética, geometría, álgebra, física y astronomía recreativas)

Martin Gardner (1914-2010)
- **"Mathematical Games"**; 288 columnas en Scientific American (1957 - 1980, 1986).
- Alicia anotada (The Annotated Alice) (1960); edición comentada por Martin Gardner de los cuentos de Lewis Carroll: Alicia en el País de las Maravillas y Alicia a través del espejo
- Ajá. Paradojas que te hacen pensar; Labor. -- otros en español:
https://www.todostuslibros.com/autor/martin-gardner

El problema de los sombreros

MATEMÁTICAS RECREATIVAS Y PROGRAMACIÓN RECREATIVA

(puzzles con diálogos)

Mayo de 1983. Sebelik, Paco y Blas se encuentran en una habitación.
Sebelik ve a Paco y a Blas, Paco ve solo a Blas; Blas no ve a nadie.
Saben que en otra habitación hay cinco sombreros: dos negros y tres blancos.
Se apaga la luz, alguien entra, toma tres sombreros y los coloca en las cabezas de S,P y B.
Tras encender la luz escuchamos el siguiente diálogo:

Sebelik.— Yo no sé el color de mi sombrero.
Paco.— En ese caso, yo tampoco.
Blas.— Ajá, ya sé el color de mi sombrero.

¿De qué color es el sombrero de Blas?

Razonando con Haskell. Un curso sobre Programación Funcional; B. Ruiz, F. Gutiérrez, P. Guerrero, J. Gallardo; Ed. Thomson (2004)

El problema de los sombreros

MATEMÁTICAS RECREATIVAS Y PROGRAMACIÓN RECREATIVA

(puzzles con diálogos)

```
data Color = B | N deriving (Eq,Show)
colores = [B,N]
posibles = [ (x,y,z) |  x<- colores,y<- colores,z<- colores,  (x,y,z)/= (N,N,N)]

quitaRep (x:xs) = x : quitaRep [ y | y<-xs, y/=x];
quitaRep []     = []

varios (_:_:_)  = True ;
varios _        = False

-- Sebelik.— Yo no sé el color de mi sombrero.
sebelik  p b   = (varios . quitaRep) [ s | (p',b',s) <-  posibles,(p',b')== (p,b) ]

-- Paco.— En ese caso, yo tampoco.
paco b    = (varios . quitaRep) [ p | p <-  colores, sebelik p b]
blas = [ b | b <-  colores, paco b ]
```

Problema 4: Cálculo del máximo y del mínimo

¿ Cuántas comparaciones debo realizar para calcular la edad máxima y la mínima de un conjunto de personas ?

Estrategia: utilizar una hoja de papel para anotar el máximo y el mínimo

```
minYmax (x:xs) = siguiente x x xs   --  anotamos x  como "candidato" a mínimo y  máximo

siguiente min max [ ] = (min,max)
siguiente min max (x:xs)
 |  x     <=   min   = siguiente x max xs     --  modifico el nuevo mínimo
 |  max >=    x      = siguiente min x xs
 |  otherwise       = siguiente min max xs   -- en este caso,  min < x < max
```

¿ Cuántas comparaciones se realizan si la lista tiene n elementos ?

al menos : __ como máximo : __ de media : __

```
*Charla> minYmax [87,100,2,99,23,98,4,97]
(2,100)
```

La conjetura de Collatz

Lothar Collatz (matemático alemán) propuso en 1937 la conjetura que lleva su nombre

Sea la función $f : \mathbb{N} \to \mathbb{N}$

$$f(n) = (3n+1)/2, \; si\ n\ es\ impar$$
$$= n/2, \qquad si\ n\ es\ par$$

Apliquemos sucesivamente desde 6:

$$6 \to f(6) = 3 \to 5 \to 8 \to 4 \to 2 \to 1 \to 2 \to 1 \ldots$$

"independientemente del número de partida la sucesión alcanzará el 1"

"Hoy día, las matemáticas no están lo suficientemente maduras para tales preguntas"

Paul Erdös (1913 – 1996)

```
fc n  | n `mod` 2 /= 0    = (3*n+1) `div` 2
      | otherwise         = n `div `2
```

```
sc n  | n==1              = [1]
      | otherwise         =  n : sc (fc n)
```

```
--  ESTUDIO DEL NÚMERO DE ITERACIONES PASOS PARA LLEGAR al 1
compruebaCollatz a b = [ n | n <- [a..b] , length (sc n) > n ]
```

```
*Charla> compruebaCollatz 1 100000
[3,6,7,9,27,31,41,47,54,55,62,63,73]
```

¡AJÁ! OTRA CONJETURA INTERESANTE