

Recuperatorio Haskell - Tema 2

Importante

Template de funciones a implementar [aca](#)
Lista de funciones permitidas [aca](#)
Ejemplo de hunit [aca](#)

Enunciado

Una **sopa de números** es un juego que consiste en descubrir propiedades de un tablero de dimensiones $n \times m$ con n y $m > 0$, en los que en cada posición hay un número entero positivo. Cada posición se identifica con una dupla (i,j) en el cual la primera componente corresponde a una fila y la segunda a una columna. A modo de ejemplo, la siguiente figura muestra un tablero de 5×4 en el que el número 13 aparece en la posición $(1,1)$ y el número 5 aparece en la posición $(4,3)$. Notar que tanto la numeración de las filas como la de las columnas comienzan en 1.

5	12	6	4
16	8	32	25
9	4	14	7
7	2	1	16
27	2	8	18

Un camino en un tablero está dado por una secuencia de posiciones adyacentes en la que solo es posible desplazarse desde una posición dada hacia la posición de su derecha o hacia la que se encuentra debajo. En otras palabras, un camino de longitud l en un tablero se define como una secuencia con l posiciones, ordenadas de manera tal que el elemento i -ésimo es la posición resultante de haberse movido hacia la derecha o hacia abajo desde la posición $(i-1)$ -ésima. Siguiendo con el ejemplo, a continuación puede observarse un camino de longitud 6 que representa la sucesión de Collatz y que empieza en la posición $(1,1)$ y termina en $(4,3)$ del tablero.

5	12	6	4
16	8	32	25
9	4	14	7
7	2	1	16
27	2	8	18

Para manipular las sopas de números en Haskell vamos a representar el tablero como una lista de filas de igual longitud. A su vez, cada fila vamos a representarla como una lista de enteros positivos. Las posiciones vamos a representarlas con tuplas de dos números enteros positivos y un camino va a estar dado por una lista de posiciones.

Para implementar esta sopa de números nos enviaron las siguientes especificaciones y nos pidieron que hagamos el desarrollo enteramente en Haskell, utilizando los tipos requeridos y solamente las funciones que se ven en la materia Introducción a la Programación / Algoritmos y Estructuras de Datos I (FCEyN-UBA).

Asumimos los siguientes renombres de tipos de datos en las especificaciones de los ejercicios:

- Fila = $\text{seq}\langle\mathbb{Z}\rangle$
- Tablero = $\text{seq}\langle\text{Fila}\rangle$

- Posicion = $Z \times Z$ -- Observación: las posiciones son: (fila, columna)
- Camino = $\text{seq}\langle \text{Posicion} \rangle$

1. Ejercicio 1 (2 puntos)

problema minimo (t : Tablero) : Z {

requiere: {El tablero t es un tablero bien formado, es decir, la longitud de todas las filas es la misma, y tienen al menos un elemento}

requiere: {Existe al menos una columna en el tablero t }

requiere: {El tablero t no es vacío, todos los números del tablero son positivos, mayor estricto a 0}

asegura: { res es igual al número más pequeño del tablero t }

}

2. Ejercicio 2 (2 puntos)

problema repetidos (t : Tablero) : $\text{seq}\langle Z \rangle$ {

requiere: {El tablero t es un tablero bien formado, es decir, la longitud de todas las filas es la misma, y tienen al menos un elemento}

requiere: {Existe al menos una columna en el tablero t }

requiere: {El tablero t no es vacío, todos los números del tablero son positivos, mayor estricto a 0}

asegura: { res es igual a la secuencia de números que aparecen al menos 2 veces en el tablero t . res no tiene repetidos.}

}

3. Ejercicio 3 (2 puntos)

problema valoresDeCamino (t : Tablero, c : Camino) : $\text{seq}\langle Z \rangle$ {

requiere: {El tablero t es un tablero bien formado, es decir, la longitud de todas las filas es la misma, y tienen al menos un elemento}

requiere: {Existe al menos una columna en el tablero t }

requiere: {El tablero t no es vacío, todos los números del tablero son positivos, mayor estricto a 0}

requiere: {El camino c es un camino válido, es decir, secuencia de posiciones adyacentes en la que solo es posible desplazarse hacia la posición de la derecha o hacia abajo y todas las posiciones están dentro de los límites del tablero t }

asegura: { res es igual a la secuencia de números que están en el camino c , ordenados de la misma forma que aparecen las posiciones correspondientes en el camino.}

}

4. Ejercicio 4 (2 puntos)

problema esCaminoCollatz (s : $\text{seq}\langle Z \rangle$, i : Z) : Bool {

requiere: {La secuencia de números s es no vacía y está compuesta por números positivos (mayor estricto que cero) que representan los valores de un camino en un tablero}

requiere: { $i \geq 0$ }

asegura: { $res = \text{true} \iff$ los valores de s son la sucesión de Collatz inicializada con el número pasado como parámetro i }

}

Notas:

En este ejercicio se pasa una secuencia de valores en lugar de un tablero y un camino para no generar dependencia con el ejercicio anterior.

La sucesión de Collatz está definida con la siguiente función recursiva:

$f(1) = 1$

$f(n) = f(n/2)$ si n es par

$f(n) = f(3n+1)$ si n es impar

Se sabe (aunque aún no está formalmente demostrado) que independientemente del n esta función siempre converge a 1 y a partir de ahí se queda ciclando (devolviendo 4, 2, 1 sucesivamente). Esto se conoce como la Conjetura de Collatz. Por ejemplo, la sucesión de Collatz para $n = 5$ es:

- **$n=5$** (impar): siguiente número es $3 * 5 + 1 = 16$.
- **$n=16$** (par): siguiente número es $16 / 2 = 8$.
- **$n=8$** (par): siguiente número es $8 / 2 = 4$.
- **$n=4$** (par): siguiente número es $4 / 2 = 2$.
- **$n=2$** (par): siguiente número es $2 / 2 = 1$.
- **$n=1$** : fin por caso base

Por lo tanto, la sucesión de Collatz para $n = 5$ es: [5, 16, 8, 4, 2, 1]

En el ejemplo del tablero y del camino (verde claro) que figuran más arriba tenemos que es `CaminoCollatz [5,16,8,4,2,1] 5` reduce a `True`.

5. Ejercicio 5 (2 puntos)

Conteste marcando la opción correcta. Haskell es un lenguaje de programación con evaluación:

- ☐ Perezosa (o *lazy*) porque una expresión solo se evalúa cuando le toca, yendo de izquierda a derecha en el código.
- ☐ Ansiosa (o *eager*) porque una expresión se evalúa tan pronto como se vincula a una variable o se pasa como un argumento.
- ☐ Perezosa (o *lazy*) porque una expresión solo se evalúa cuando se necesita su valor.

Adjunta el archivo con tu solución:

Solo se puede adjuntar 1 archivo de extensión .hs. En caso de haber desarrollado tests propios, no deben ser entregados.

Seleccionar archivo a enviar: No file chosen

Enviar