Trabajo Práctico

Contando círculos primos

Taller de Álgebra
Segundo cuatrimestre 2019

Fecha límite de entrega:
Domingo 17 de noviembre hasta las **23:59** hs.
Coloquio: 20 y 22 de noviembre (en el turno del grupo)

Introducción

Un **círculo de orden** n es una permutación circular de los números del 1 al n. Dos círculos son iguales si difieren por rotación. Por ejemplo, los dos primeros círculos de la la Figura 1 son iguales, mientras que el tercero es distinto a ellos.

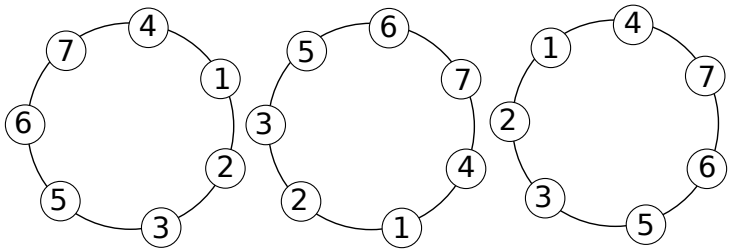

Figura 1: Círculos de orden 7

Un **círculo primo de orden** n es un círculo de orden n donde todos los pares de números adyacentes suman un número primo. A continuación se presentan a modo de ejemplo algunos cículos primos.

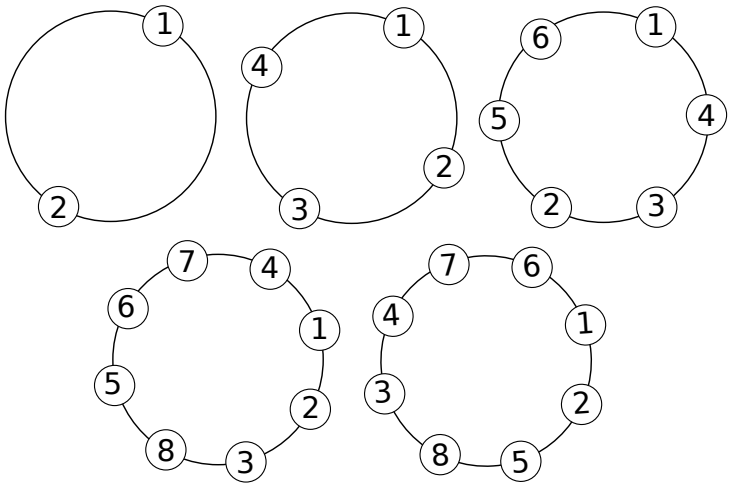

Figura 2: Círculos primos de orden 2, 4, 6 y 8

En este Trabajo Práctico se pide escribir funciones que cuenten en forma exhaustiva todos los círculos primos de un tamaño dado.

Vamos a representar un círculo de orden n como una lista de longitud n cuyos elementos son todos los elementos del círculo, listados en sentido horario y comenzando en cualquier lugar. Por ejemplo, el primer círculo primo de la izquierda de la segunda fila de la Figura 2 se puede representar mediante la lista `[7,4,1,2,3,8,5,6]`, o también mediante la lista `[6,7,4,1,2,3,8,5]`.

Ejercicios obligatorios

Escribir las siguientes funciones en Haskell, definiendo el tipo `Circulo` como lista de enteros.

1. `sonCirculosIguales :: Circulo -> Circulo -> Bool`
   `sonCirculosIguales circulo1 circulo2`
   que determine si `circulo1` y `circulo2` son iguales (teniendo en cuenta que dos círculos son iguales si difieren por rotación, y además que cada círculo puede representarse mediante más de una lista).

   Por ejemplo, los círculos `[7,4,1,2,3,8,5,6]` y `[6,7,4,1,2,3,8,5]` son iguales.

   Aclaraciones:

   - `circulo1` y `circulo2` pueden tener distinta cantidad de elementos. En esos casos debe devolver `False`

2. `permutaciones :: Integer -> [[Integer]]`
   `permutaciones n`

   que genere una lista con todas las permutaciones de los enteros entre `1` y `n` como listas de enteros.

   Por ejemplo:

   `permutaciones 3` debe resultar `[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]`, u otra lista que pueda formarse con estas 6 permutaciones en otro orden.

3. `esCirculoPrimo :: Circulo -> Bool`
   `esCirculoPrimo circulo`

   que determine si `circulo` cumple las condiciones necesarias para ser un círculo primo. Se puede asumir que el círculo es de orden mayor a 1.

4. `estaRepetidoPrimero :: [Circulo] -> Bool`
   `estaRepetidoPrimero circulos`

   que determine si el primer círculo de la lista `circulos` está repetido en otro lugar de esa lista.

5. `listaCirculosPrimos :: Integer -> [Circulo]`
   `listaCirculosPrimos n`

   que, dado $n \geq 2$, devuelve una lista de todos los círculos primos distintos de longitud `n`.

6. `contarCirculosPrimos :: Integer -> Integer`
   `contarCirculosPrimos n`

   que devuelve la cantidad de círculos primos distintos de orden `n`, para $n \geq 2$.

Ejercicio optativo

1. `listaCirculosPrimosEspejados :: Integer -> [Circulo]`
   `listaCirculosPrimosEspejados n`
   que devuelve la lista de círculos primos espejados distintos de orden `n`, para $n \geq 2$. Dos círculos son el mismo círculo espejado si difieren por simetría (podrían, además, diferir por rotación). Por ejemplo, `[6,7,4,1,2,3,8,5]`, `[6,5,8,3,2,1,4,7]` definen el mismo círculo primo espejado.

Pautas de Entrega

- El trabajo se debe realizar en grupos de tres alumnos.
- El archivo con el código fuente debe tener nombre `nrogrupo_alg1-tp.hs` (por ej. para el grupo 3 el nombre debe ser `3_alg1-tp.hs`), y debe entregarse mediante el formulario `http://cor.to/TP-algeb2019-c2`[1]. Además, en el archivo entregado debe indicarse, en un comentario arriba de todo: nombre y LU (o DNI) de cada integrante.
- El código debe poder ser ejecutado en el GHCI instalado en los laboratorios del DC, sin ningún paquete especial.
- No está permitido alterar los tipos de datos presentados en el enunciado, ni utilizar técnicas no vistas en clase para resolver los ejercicios (como por ejemplo, alto orden).
- Pueden definirse todas las funciones auxiliares que se requieran. Cada una debe tener un comentario indicando claramente qué hace.
- No es necesario entregar un informe sobre el trabajo.
- La fecha límite de entrega es el domingo 17/11 a las 23:59.

Los objetivos a evaluar en el código de este trabajo práctico son:

- Resolución correcta.
- Declaratividad.
- Prolijidad: evitar repetir código innecesariamente y usar adecuadamente las funciones previamente definidas (tener en cuenta tanto las funciones definidas en el enunciado como las definidas por ustedes mismos).
- Uso adecuado de las técnicas vistas en clase como recursión o pattern matching.

Importante: se admitirá un único envío, sin excepción, por grupo. Planifiquen el trabajo para llegar a tiempo con la entrega.

Referencias del lenguaje Haskell

- **The Haskell 2010 Language Report:** la última versión oficial del lenguaje Haskell a la fecha, disponible online en `http://www.haskell.org/onlinereport/haskell2010`.
- **Learn You a Haskell for Great Good!**: libro accesible, para todas las edades, cubriendo todos los aspectos del lenguaje, notoriamente ilustrado, disponible online en `http://learnyouahaskell.com` y en `http://aprendehaskell.es/` (en español).
- **Real World Haskell**: libro apuntado a zanjar la brecha de aplicación de Haskell, enfocándose principalmente en la utilización de estructuras de datos funcionales en la "vida real", disponible online en `http://book.realworldhaskell.org/read`.
- **Hoogle**: buscador que acepta tanto nombres de funciones y módulos, como signaturas y tipos *parciales*, online en `http://www.haskell.org/hoogle/`

[1] Dirección completa: `https://forms.gle/K3yGVAN1BwoaLrqv8`