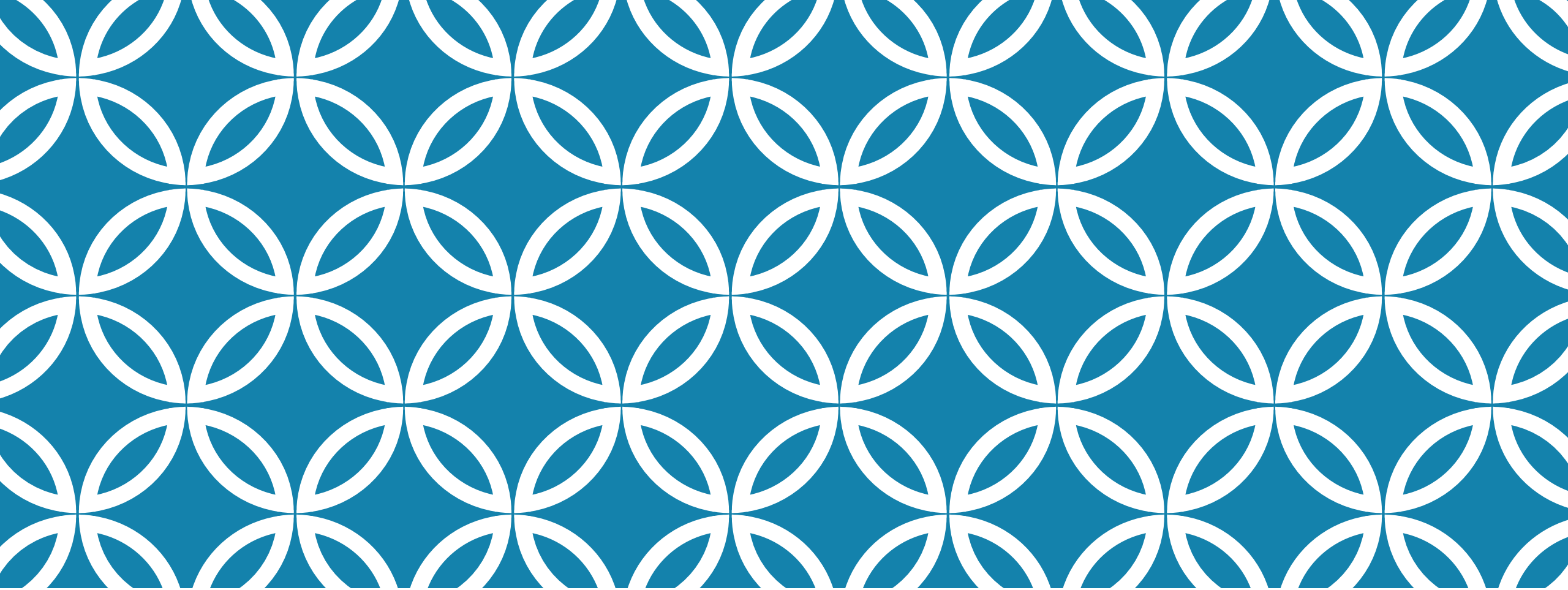

ESTRUCTURAS DISCRETAS

Estructuras de Datos recursivas e Inducción

Javier Enríquez Mendoza

Mauricio E. Hernández Olvera

ESTRUCTURAS DE DATOS

¿ Qué es una Estructura de Datos?

Una estructura de Datos es una forma de organizar información para poder manipularla y trabajar con ella de forma eficiente.

Hasta ahora en Haskell hemos trabajado únicamente con 2 estructuras :

- Listas
- Tuplas

Ambas ya están predefinidas en el núcleo de Haskell.

ESTRUCTURAS DE DATOS

Existen muchos tipos de estructuras de datos que como programadores nos resultan muy útiles. Por ejemplo:

- Arrays
- Listas
- Tuplas
- Arboles Binarios
- Pilas
- Colas
- Árboles

Entre muchas otras.

LISTAS

Son la estructura de datos mas común en programación funcional.

Pueden almacenar varios elementos

En Haskell son Homogéneas.

Están definidas de la siguiente manera:

- La lista vacía es una lista.
- Sean x un elemento de tipo A y xs una lista de elementos de tipo A entonces (Cons x xs) es una lista.
- Estas son todas.

IMPLEMENTACIÓN EN HASKELL

Se puede traducir fácilmente la definición formal de listas al lenguaje de programación Haskell de la siguiente forma:

```
data List a = Empty | Cons a (List a)
```

Que es muy parecida a la definición de listas que tiene Haskell.

Para las listas que son construidas con el constructor Cons de la siguiente forma

- Cons x xs
- x es la cabeza de la lista.
- xs es la cola de la lista

Esta definición a pesar de ser la mas común, no es la única que existe.

LISTAS SNOC

La definición de este tipo de **listas** es la siguiente:

- la lista vacia es una lista.
- Sean xs una lista de elementos de tipo A y x un elemento de tipo A, entoncesn (Snoc xs a) es una lista.
- Estas son todas.

En donde la operación **Snoc** pega el elemento x como último elemento a la lista xs.

- x es el último elemento.
- xs es el resto de la lista.

De igual manera se puede traducir muy fácilmente de la siguiente forma:

```
data ListSnoc a = Mt | Snoc (ListSnoc a) a
```

ÁRBOLES BINARIOS

Los **árboles binarios** son estructuras de datos que se definen de la siguiente forma:

- Un árbol vacío es un árbol binario.
- Si T_1 y T_2 son árboles binarios y c es un elemento de A, entonces (Node T_1 c T_2) es también un árbol binario. Donde T_1 es el subárbol izquierdo y T_2 es el subárbol derecho. Al nodo c se le llama raíz.
- Nada más es un árbol binario

Esta definición puede traducirse a un tipo de dato algebraico en Haskell.

```
data BinaryTree a = Void | Node (BinaryTree a) a (BinaryTree a)
```

ÁRBOLES BINARIOS ORDENADOS

Definición

Un árbol Binario esta **ordenado** si y sólo si para todo nodo (Node T_1 r T_2) en el árbol se cumplen las siguientes condiciones:

- r es mayor a todos los elementos de T_1 o T_1 es vacío
- r es menor a todos los elementos de T_2 o T_2 es vacío

¿Por qué importa el orden?

Mejorar la eficiencia en la búsqueda de elementos.

Al recorrer el árbol obtenemos los elementos en orden ascendente.

RECORRIDOS

Preorder:

- Se visita la raíz
- Se visita el subárbol izquierdo
- Se visita el subárbol derecho

Inorder:

- Se visita el subárbol izquierdo
- Se visita la raíz
- Se visita el subárbol derecho

Postorder

- Se visita el subárbol izquierdo
- Se visita el subárbol derecho
- Se visita la raíz

BÚSQUEDA

Para buscar un elemento en un árbol binario usaremos el siguiente algoritmo

1. Si el árbol es vacío, no encontramos el elemento y terminamos
2. Verificamos si el elemento que buscamos es la raíz. De ser así ya lo encontramos, terminamos.
3. Si el elemento es menor a la raíz, buscamos recursivamente en el subárbol izquierdo.
4. Si el elemento es mayor a la raíz, buscamos recursivamente en el subárbol derecho.

LOS NÚMEROS NATURALES

Para definir el conjunto de los números naturales, se usan los axiomas de Peano.

- 0 es un natural
- Si n es un natural entonces $S\ (n)$ es un natural
- $\forall\, n\ (S\ (n)\ \neq 0)$
- $\forall\, n \forall\, m\ \Big(\big(S\ (n) = S\ (m)\big)\ \rightarrow\ (n = m)\Big)$

Los expresamos en Haskell de la siguiente forma

```
data Nat = Cero | S Nat
```

INDUCCIÓN SOBRE NATURALES

La inducción consiste en demostrar proposiciones sobre un conjunto, verificando casos particulares.

Se demuestra la proposición para el conjunto de casos base.

Se supone que la propiedad se cumple para n.

Se demuestra que la proposición se cumple para la regla recursivas $S\ (n)$.

INDUCCIÓN ESTRUCTURAL

Base Inductiva

Si a es un elemento de A generado por una regla básica, entonces debemos probar directamente la validez de P(a).

Si x es un elemento de A construido mediante alguna regla recursiva a partir de elementos anteriores $x_1, \ldots, x_n$ entonces procedemos como sigue:

Hipótesis Inductiva: Suponer $P(x_1), \ldots, P(x_n)$.

Paso Inductivo: Probar P(x).

En este caso, el principio de induccion´ estructural permite concluir que $\forall x P(x)$.

Toda definición recursiva genera un principio de Inducción