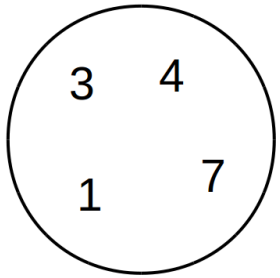
Taller de Álgebra I

Clase 7 - Conjuntos

Segundo cuatrimestre 2022

Conjuntos

Supongamos que queremos representar un **conjunto** de números enteros.

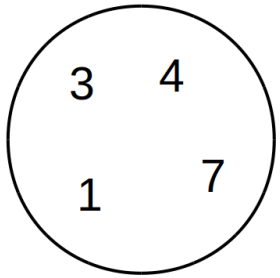


¿Es buena idea usar una lista `[Int]`?

- Podríamos representar ese conjunto con la lista `[1,3,4,7]`.

Conjuntos

Supongamos que queremos representar un **conjunto** de números enteros.

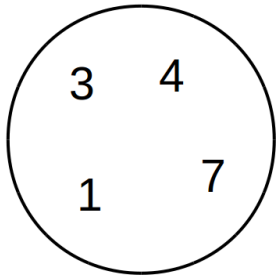


¿Es buena idea usar una lista `[Int]`?

- Podríamos representar ese conjunto con la lista `[1,3,4,7]`.
  - También con `[4,1,3,7]`, `[3,7,4,1]`, `[7,3,1,4]`, ...
  - Todas estas listas son **distintas**, pero representan al **mismo** conjunto.
  - El orden de los elementos es relevante para las listas, pero no para conjuntos.

Conjuntos

Supongamos que queremos representar un **conjunto** de números enteros.

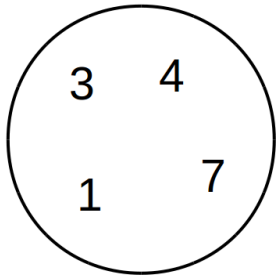


¿Es buena idea usar una lista `[Int]`?

- Podríamos representar ese conjunto con la lista `[1,3,4,7]`.
  - También con `[4,1,3,7]`, `[3,7,4,1]`, `[7,3,1,4]`, ...
  - Todas estas listas son **distintas**, pero representan al **mismo** conjunto.
  - El orden de los elementos es relevante para las listas, pero no para conjuntos.
- ¿Y la lista `[1,3,4,7,7,7,1,4,7]`? ¿Sirve para representar a nuestro conjunto?

Conjuntos

Supongamos que queremos representar un **conjunto** de números enteros.

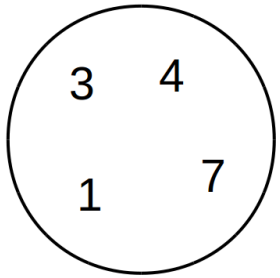


¿Es buena idea usar una lista `[Int]`?

- Podríamos representar ese conjunto con la lista `[1,3,4,7]`.
  - También con `[4,1,3,7]`, `[3,7,4,1]`, `[7,3,1,4]`, ...
  - Todas estas listas son **distintas**, pero representan al **mismo** conjunto.
  - El orden de los elementos es relevante para las listas, pero no para conjuntos.
- ¿Y la lista `[1,3,4,7,7,7,1,4,7]`? ¿Sirve para representar a nuestro conjunto?
  - Las listas pueden tener elementos repetidos, pero eso no tiene sentido con conjuntos.

Conjuntos

Supongamos que queremos representar un **conjunto** de números enteros.



¿Es buena idea usar una lista `[Int]`?

- Podríamos representar ese conjunto con la lista `[1,3,4,7]`.
  - También con `[4,1,3,7]`, `[3,7,4,1]`, `[7,3,1,4]`, ...
  - Todas estas listas son **distintas**, pero representan al **mismo** conjunto.
  - El orden de los elementos es relevante para las listas, pero no para conjuntos.
- ¿Y la lista `[1,3,4,7,7,7,1,4,7]`? ¿Sirve para representar a nuestro conjunto?
  - Las listas pueden tener elementos repetidos, pero eso no tiene sentido con conjuntos.

Vamos a usar `[Int]` para representar conjuntos, pero dejando claro que hablamos de conjuntos (sin orden ni repetidos). Para eso podemos hacer un renombre de tipos.

Definición de tipo usando `type`

Definamos un renombre de tipos para conjuntos: `type Set a = [a]`

- Otra forma de escribir lo mismo, pero más descriptivo.
- `type` es la palabra reservada del lenguaje, `Set` es el nombre que le pusimos nosotros.
- Si bien internamente es una lista, la idea es tratar a `Set a` como si fuera conjunto (es un contrato entre programadores).
- Si nuestra función recibe un conjunto, **vamos a suponer** que no contiene elementos repetidos. (Haskell no hace nada para verificarlo.)
- Si nuestra función devuelve un conjunto, **debemos asegurar** que no contiene elementos repetidos. (Haskell tampoco hace nada automático.)
- Además, no hace falta preocuparse por el orden de los elementos. (Haskell no lo sabe.)

Conjuntos

Definición de tipo usando `type`

Definamos un renombre de tipos para conjuntos: `type Set a = [a]`

- Otra forma de escribir lo mismo, pero más descriptivo.
- `type` es la palabra reservada del lenguaje, `Set` es el nombre que le pusimos nosotros.
- Si bien internamente es una lista, la idea es tratar a `Set` a como si fuera conjunto (es un contrato entre programadores).
- Si nuestra función recibe un conjunto, **vamos a suponer** que no contiene elementos repetidos. (Haskell no hace nada para verificarlo.)
- Si nuestra función devuelve un conjunto, **debemos asegurar** que no contiene elementos repetidos. (Haskell tampoco hace nada automático.)
- Además, no hace falta preocuparse por el orden de los elementos. (Haskell no lo sabe.)

Ejercicios entre todos

- Definir `vacio :: Set Int` que represente el conjunto vacío
- Implementar entre todos la función
  `agregar :: Int -> Set Int -> Set Int`
  que dado un entero y un conjunto agrega el primero al segundo (ayuda: La función "pertenece" en Haskell existe y se llama "elem")

Conjuntos

Ejercicios simples

1. Implementar una función `incluido :: Set Int -> Set Int -> Bool` que determina si el primer conjunto está incluido en el segundo.
2. Implementar una función `iguales :: Set Int -> Set Int -> Bool` que determina si dos conjuntos son iguales.

Partes de un conjunto

Ejercicios

1. Implementar una función
   `partes :: Int -> Set (Set Int)` que genere todos los subconjuntos del conjunto $\{1, 2, 3, \ldots, n\}$.

```
Ejemplo> partes 2
[[], [1], [2], [1, 2]]
```

Producto Cartesiano

Producto cartesiano

- Implementar una función `productoCartesiano :: Set Int -> Set Int -> Set (Int, Int)` que dados dos conjuntos genere todos los pares posibles (como pares de dos elementos) tomando el primer elemento del primer conjunto y el segundo elemento del segundo conjunto.

```
Ejemplo> productoCartesiano [1, 2, 3] [3, 4]
[(1, 3), (2, 3), (3, 3), (1, 4), (2, 4), (3, 4)]
```

- ¿Cómo podemos encarar este ejercicio?

Producto Cartesiano

Producto cartesiano

- Implementar una función `productoCartesiano :: Set Int -> Set Int -> Set (Int, Int)` que dados dos conjuntos genere todos los pares posibles (como pares de dos elementos) tomando el primer elemento del primer conjunto y el segundo elemento del segundo conjunto.

```
Ejemplo> productoCartesiano [1, 2, 3] [3, 4]
[(1, 3), (2, 3), (3, 3), (1, 4), (2, 4), (3, 4)]
```

- ¿Cómo podemos encarar este ejercicio?
- Notar que tenemos dos parámetros sobre los que tenemos que hacer recursión para obtener todos los pares.

Producto Cartesiano

Producto cartesiano

- Implementar una función `productoCartesiano :: Set Int -> Set Int -> Set (Int, Int)` que dados dos conjuntos genere todos los pares posibles (como pares de dos elementos) tomando el primer elemento del primer conjunto y el segundo elemento del segundo conjunto.

```
Ejemplo> productoCartesiano [1, 2, 3] [3, 4]
[(1, 3), (2, 3), (3, 3), (1, 4), (2, 4), (3, 4)]
```

- ¿Cómo podemos encarar este ejercicio?
- Notar que tenemos dos parámetros sobre los que tenemos que hacer recursión para obtener todos los pares.
- Podría servir alguna idea como la de la suma doble...