Taller de Álgebra I

Clase 8 - Combinatoria

Número Combinatorio

Ejercicio

- Escribir una función que dados n, $k \in \mathbb{N}$ tal que $0 \leq k \leq n$, compute el combinatorio $\binom{n}{k}$.

Número Combinatorio

Ejercicio

- Escribir una función que dados n, $k \in \mathbb{N}$ tal que $0 \leq k \leq n$, compute el combinatorio $\binom{n}{k}$.
  - Hacerlo usando la igualdad $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ para $1 \leq k \leq n-1$

Variaciones con repetición

- Implementar una función
  `variaciones :: Set Int -> Int -> Set [Int]` que dado un conjunto c y una longitud k genere todas las posibles listas de longitud k a partir de elementos de c.

```
Ejemplo> variaciones [4, 7] 3
[[4, 4, 4], [4, 4, 7], [4, 7, 4], [4, 7, 7], [7, 4, 4], [7, 4, 7], [7,
7, 4], [7, 7, 7]]
```

- ¿Cómo podemos pensar este ejercicio recursivamente?

Variaciones con repetición

- Implementar una función
  `variaciones :: Set Int -> Int -> Set [Int]` que dado un conjunto c y una longitud k genere todas las posibles listas de longitud k a partir de elementos de c.

```
Ejemplo> variaciones [4, 7] 3
[[4, 4, 4], [4, 4, 7], [4, 7, 4], [4, 7, 7], [7, 4, 4], [7, 4, 7], [7,
7, 4], [7, 7, 7]]
```

- ¿Cómo podemos pensar este ejercicio recursivamente?

Variaciones con repetición

- Implementar una función
  `variaciones :: Set Int -> Int -> Set [Int]` que dado un conjunto c y una longitud k genere todas las posibles listas de longitud k a partir de elementos de c.

```
Ejemplo> variaciones [4, 7] 3
[[4, 4, 4], [4, 4, 7], [4, 7, 4], [4, 7, 7], [7, 4, 4], [7, 4, 7], [7,
7, 4], [7, 7, 7]]
```

- ¿Cómo podemos pensar este ejercicio recursivamente?

```
- variaciones [4, 7]  0 = [[]]
```

Variaciones con repetición

- Implementar una función
  `variaciones :: Set Int -> Int -> Set [Int]` que dado un conjunto c y una longitud k genere todas las posibles listas de longitud k a partir de elementos de c.

```
Ejemplo> variaciones [4, 7] 3
[[4, 4, 4], [4, 4, 7], [4, 7, 4], [4, 7, 7], [7, 4, 4], [7, 4, 7], [7,
7, 4], [7, 7, 7]]
```

- ¿Cómo podemos pensar este ejercicio recursivamente?

```
- variaciones [4, 7]  0 = [[]]
- variaciones [4, 7]  1 = [[4], [7]]
```

Variaciones con repetición

- Implementar una función
  `variaciones :: Set Int -> Int -> Set [Int]` que dado un conjunto c y una longitud k genere todas las posibles listas de longitud k a partir de elementos de c.

```
Ejemplo> variaciones [4, 7] 3
[[4, 4, 4], [4, 4, 7], [4, 7, 4], [4, 7, 7], [7, 4, 4], [7, 4, 7], [7,
7, 4], [7, 7, 7]]
```

- ¿Cómo podemos pensar este ejercicio recursivamente?

```
- variaciones [4, 7]  0 = [[]]
- variaciones [4, 7]  1 = [[4], [7]]
- variaciones [4, 7]  2 = [[4, 4], [4, 7], [7, 4], [7, 7]]
```

Permutaciones

Insertar un elemento en una lista

- Implementar una función `insertarEn :: [Int] -> Int -> Int -> [Int]` que dados una lista *l*, un número *n* y una posición *i* (contando desde 1) devuelva una lista en donde se insertó *n* en la posición *i* de *l* y los elementos siguientes corridos en una posición.

```
Ejemplo> insertarEn [1, 2, 3, 4, 5] 6 2
[1, 6, 2, 3, 4, 5]
```

Permutaciones

Insertar un elemento en una lista

- Implementar una función
  `insertarEn :: [Int] -> Int -> Int -> [Int]` que dados una lista *l*, un número *n* y una posición *i* (contando desde 1) devuelva una lista en donde se insertó *n* en la posición *i* de *l* y los elementos siguientes corridos en una posición.

```
Ejemplo> insertarEn [1, 2, 3, 4, 5] 6 2
[1, 6, 2, 3, 4, 5]
```

Permutaciones

- Implementar una función

  `permutaciones :: Set Int -> Set [Int]`

  que dado un conjunto de enteros, genere todas las posibles permutaciones de los números del conjunto pasado por parámetro.

```
Ejemplo> permutaciones [1,2,3]
[[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]
```

Ejercicios

Implementar funciones que devuelvan

1. Todas las formas de ubicar n bolitas numeradas en k cajas.
   `bolitasEnCajas :: Int -> Int -> Set [Int]`

   ```
   Ejemplo> bolitasEnCajas 2 3
   [[1,1],[1,2],[1,3],[2,1],[2,2],[2,3],[3,1],[3,2],[3,3]]
   ```

   Notar que el elemento i de cada sublista representa el número de caja donde fue a parar la bolita i.

2. Todas las formas de ubicar n bolitas numeradas en k cajas tal que la primera caja nunca esté vacía.

3. Todas las listas ordenadas de k números distintos tomados del conjunto $\{1, \ldots, n\}$.

4. Todas las sucesiones de los caracteres 'a' y 'b' de longitud n y m respectivamente.

5. Todas las sucesiones de 'a', 'b' y 'c' de longitud n, m y k respectivamente.

6. Implementar una función
   `subconjuntos :: Set Int -> Int -> Set (Set Int)` que dados un conjunto de enteros y un entero k, genera todos los subconjuntos de k elementos del conjunto pasado por parámetro.

   ```
   Ejemplo> subjconjuntos [1,2,3] 2
   [[1, 2], [2, 3], [1, 3]]
   ```