Práctica Evaluable - Noviembre 2020

Estructuras de Datos

Computadores A / Informática D / Matemáticas + Informática

Definición informal del TAD

El TAD SetMultiMap es una colección de pares `(K,Vs)` donde:

- `K` es la *clave* y
- `Vs` un *conjunto no vacío* de valores de tipo `V` asociado a la clave `K`.

En un SetMultiMap las claves son **únicas** (no puede haber dos pares con la misma clave).

Ejemplo. El siguiente es un SetMultiMap que asocia a cada clave (`String`) un conjunto de enteros (`Int`) (representamos los pares `(K,Vs)` por la notación gráfica `K -> { V1, ..., Vn }`):

```
"alfredo" -> { 9 }
"juan" -> { 8, 1, 0 }
"maria" -> { 4, -6, 8 }
```

Un SetMultiMap `xs` soporta las siguientes operaciones:

- `empty`: devuelve un SetMultiMap vacío.
- `isEmpty xs`: devuelve `True` si `xs` está vacío, `False` en caso contrario.
- `size xs`: devuelve el número de pares que contiene `xs`.
- `isDefinedAt k xs`: que devuelve `True` si la clave `k` aparece en `xs` y `False` en caso contrario.
- `insert xs k v`: si la clave `k` no está presente en `xs`, añade el par `(k, { v })`; si la clave `k` está presente en `xs`, añade `v` al conjunto de valores asociado a `k`.
- `valuesOf k xs`: devuelve, *si es posible*, el conjunto de valores asociado a `k` en `xs`.
- `deleteKey k xs`: si la clave `k` aparece en `xs` la elimina; de lo contrario no hace nada.
- `deleteKeyValue k v xs`: si la clave `k` aparece en `xs` y tiene asociado el valor `v`, elimina `v`; de lo contrario no hace nada.
- `filterValues p xs`: devuelve el SetMultimap que se obtiene al filtrar los valores asociados a cada clave con el predicado `p`.

Representación física

El TAD SetMultiMap se representará en Haskell por el siguiente tipo algebraico:

```haskell
data SetMultiMap a b = Empty
                     | Node a (S.Set b) (SetMultiMap a b)
                     deriving Eq
```

Además, se debe satisfacer el siguiente **invariante de representación**:

- Los nodos están ordenados por clave
- No hay claves repetidas
- No hay claves que tengan asociado un conjunto vacío

Ejercicios

El fichero `SetMultiMap.hs` contiene definiciones incompletas de la implementación y ejemplos de uso de cada función en los que se muestra la salida esperada para un SetMultimap `m1`.

Ejercicio 1. Completa en `SetMultiMap.hs` las definiciones de las funciones `empty`, `isEmpty`, `size`, `isDefinedAt`, `insert`, `valuesOf`, `deleteKey`, `deleteKeyValue`, y `filterValues`. Completa además el comentario indicando la clase de complejidad a la que pertenece cada función.

Ejercicio 2. Completa en `SetMultiMap.hs` los axiomas que definen la semántica de `deleteKeyValue`.

Ejercicio 3. Además de las operaciones anteriores, el TAD SetMultiMap tiene definido el siguiente plegado:

`fold f z xs`: pliega un SetMultiMap usando `z` para el caso base y `f` como función de plegado.

La función de plegado `f` recibe como parámetros la clave `k` y **cada uno** de los valores `v` asociados a `k` por separado; por ejemplo, para el SetMultiMap `m1` se `f` invocará 7 veces.

Completa en `SetMultiMapClient.hs` la definición de la función `compose`:

- `compose xs ys`: devuelve la composición de `xs` e `ys` (consultar en `SetMultiMapClient.hs` un ejemplo de uso).

Entrega de la práctica evaluable

- Se deben entregar los ficheros:
  - `SetMultiMap.hs` y
  - `SetMultiMapClient.hs`
- No olvides completar tu nombre, apellidos y grupo
- Para que una función puntúe es necesario que compile