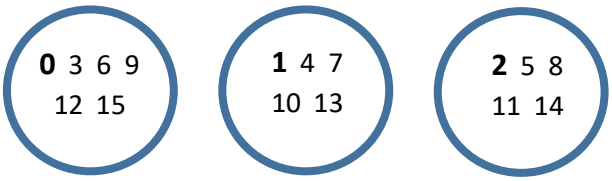
Nombre: Grupo:

Estructuras de Datos. Febrero 2018. Universidad de Málaga.

De acuerdo a la Wikipedia, en teoría de conjuntos y álgebra, la noción de *relación de equivalencia* sobre un conjunto de elementos permite establecer una relación entre los elementos del conjunto que comparten cierta característica o propiedad. Esto permite agrupar dichos elementos en *clases de equivalencia*, es decir, "paquetes" de elementos similares.

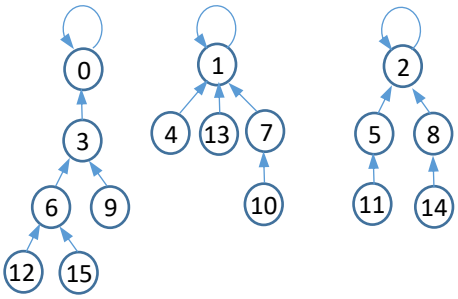
El objetivo de este ejercicio es implementar una estructura de datos, denominada `DisjointSet a`, que permita representar una relación de equivalencia. Una relación de equivalencia estará formada por una colección de clases de equivalencia, siendo cada clase un conjunto formado por elementos de tipo a. Para cada clase, uno de sus elementos será, de algún modo, el que represente a la clase; denominaremos a éste valor como el *elemento canónico* de la clase. Además, para una relación de equivalencia, un elemento concreto sólo puede estar incluido en una de las clases.

Supongamos el conjunto formado por los números naturales desde el 0 al 15 y la relación de equivalencia sobre este conjunto **módulo 3**, de forma que dos valores *x* e *y* estarán en la misma clase de equivalencia si *x* módulo 3 = *y* módulo 3. Gráficamente, las tres clases de equivalencia para esta relación serían (los elementos canónicos se muestran en negrita):



Implementaremos una clase de equivalencia `DisjointSet a` usando un diccionario (también denominado mapa) de claves de tipo a en valores de tipo a. El elemento canónico de la clase estará relacionado en el diccionario consigo mismo. Cualquier otro elemento de la clase estará relacionado con otro elemento distinto de la misma clase (no necesariamente el canónico). Por ejemplo, la relación **módulo 3** podría quedar representada con el diccionario que aparece a la izquierda de la siguiente figura, donde cada elemento delante de una fecha es una clave y cada elemento tras la flecha es su valor asociado. Este diccionario representa las relaciones mostradas a la derecha de la figura:

```
{  0 → 0,  1 → 1,  2 → 2
,  3 → 0,  4 → 1,  5 → 2
,  6 → 3,  7 → 1,  8 → 2
,  9 → 3, 10 → 7, 11 → 5
, 12 → 6, 13 → 1, 14 → 8
, 15 → 6
}
```



Haskell

Sea la siguiente definición del tipo `DisjointSet a` para representar una relación de equivalencia en Haskell:

```
import qualified DataStructures.Dictionary.AVLDictionary as D
data DisjointSet a = DS (D.Dictionary a a)
```

Implementa las siguientes operaciones para esta estructura de datos:

1) (0.25 puntos) `empty`, que devuelve una relación de equivalencia vacía, es decir, sin ningún elemento y formada por cero clases de equivalencia.

2) (0.25 puntos) `isEmpty`, que devuelve `True` si una relación de equivalencia es vacía y `isElem` que devuelve `True` si el argumento pertenece a la relación de equivalencia.

3) (0.25 puntos) `numElements`, que devuelve el número total de elementos de la relación de equivalencia, es decir, la suma del cardinal de cada clase de equivalencia en la relación (16 para la relación **módulo 3** del ejemplo).

4) (1 punto) `add`, que toma un elemento `x` y una relación de equivalencia `ds`, y devuelve una nueva relación de equivalencia, idéntica a la dada, pero incluyendo una clase adicional formada tan solo por el elemento `x`. Si elemento `x` ya estuviese previamente en la relación `ds`, se devolverá la misma relación.
5) (1.25 puntos) `root`, que toma un elemento `x` y una relación de equivalencia `ds`, y devuelve el elemento canónico de la clase a la que pertenece `x`. El resultado será de tipo `Maybe a`, de forma que si `x` no pertenece a ninguna clase se devolverá `Nothing`; se devolverá `Just r` si `x` pertenece a alguna clase y `r` es su elemento canónico.
6) (1 punto) `isRoot`, que toma un elemento `x` y una relación de equivalencia `ds`, y devuelve `True` si `x` es el elemento canónico de su clase.
7) (1 punto) `areConnected`, que toma dos elementos y una relación de equivalencia, y devuelve `True` si ambos pertenecen a la misma clase de equivalencia.
8) (1.25 puntos) `kind`, que toma un elemento `x` y una relación de equivalencia `ds`, y devuelve una lista con todos los elementos que están en la misma clase de equivalencia que `x`.
9) (1.25 puntos) `union`, que toma dos elementos y fusiona las clases de equivalencia de ambos en una única clase, devolviendo la nueva relación de equivalencia resultante. Para ello, de los elementos canónicos de ambas clases, se seleccionará el de mayor valor y se asociará en el diccionario con el canónico de menor valor (la clave es el de mayor valor).

Alumnos Tiempo Parcial
10 (1.25 puntos) `flatten`, que toma una relación de equivalencia y aplana la representación de la estructura, de manera que cada elemento se relacione con su canónico.
11 (1.25 puntos) `kinds`, que toma una relación de equivalencia y devuelve una lista con todas sus clases de equivalencia, cada una representada mediante una lista con sus elementos.

Java

Completar la clase `DisjointSetDictionary<T>` que contiene un diccionario con claves y valores de tipo `T`. Implementa los siguientes constructores y métodos para esta clase:

1) (0.25 puntos) Un constructor que inicialice adecuadamente la estructura como una clase de equivalencia vacía.
2) (0.25 puntos) El método `public boolean isEmpty()`, que devuelve `true` si una relación de equivalencia es vacía y `public boolean isElem(T elem)` que devuelve `true` si elem pertenece a la relación de equivalencia.
3) (0.25 puntos) El método `public int numElements()`, que devuelve el número total de elementos de la relación de equivalencia, es decir, la suma del cardinal de cada clase de equivalencia en la relación (16 para la relación **módulo 3** del ejemplo).
4) (1 punto) El método `public void add(T x)`, que toma un elemento `x` y modifica la relación de equivalencia incluyendo una clase adicional formada tan solo por el elemento `x`. Si elemento `x` ya estuviese en la relación, la relación no cambia.
5) (1.25 puntos) El método `private T root(T x)`, que toma un elemento `x` y devuelve el elemento canónico de la clase a la que pertenece `x`. El resultado será `null` si `x` no pertenece a ninguna clase.
6) (1 punto) El método `private boolean isRoot(T x)`, que toma un elemento `x` y devuelve `true` si `x` es el elemento canónico de su clase.
7) (1 punto) El método `public boolean areConnected(T x, T y)`, que toma dos elementos y devuelve `true` si ambos pertenecen a la misma clase de equivalencia.
8) (1.25 puntos) El método `public List<T> kind(T x)`, que toma un elemento `x` y devuelve una lista con todos los elementos que están en la misma clase de equivalencia que `x`.
9) (1.25 puntos) El método `public void union(T x, T y)`, que toma dos elementos y fusiona las clases de equivalencia de ambos en una única clase. Para ello, de los elementos canónicos de ambas clases, se seleccionará el de mayor valor y se asociará en el diccionario con el canónico de menor valor (la clave es el de mayor valor).

Alumnos Tiempo Parcial
10 (1.25 puntos) El método `public void flatten()`, que aplana la representación de la estructura, de manera que cada elemento se relacione con su canónico.
11 (1.25 puntos) El método `public List<List<T>> kinds()`, que devuelve una lista con todas las clases de equivalencia de la relación, cada una representada mediante una lista con sus elementos.