# Práctica 5: Diccionarios

**Ejercicio 1:** Aumentar la funcionalidad de las clases `LinkedBinarySearchTree`, `AVLTree` y `RBTree` de manera que sean capaces de obtener el menor (`first`)/mayor (`last`) valor dentro de un árbol, y una colección con los valores menores (`predecessors`)/mayores (`successors`) que un valor dado. Para ello hemos añadido los métodos correspondientes a la interfaz `BinarySearchTree` y a cada una de las clases que la implementan.

**Ejercicio 2:** Aumentar la funcionalidad de los diccionarios ordenados implementando el método `findRange` que aparece definido en la clase `AbstractTreeOrderedDict`. Este método permite consultar qué entradas almacenadas presentan claves cuyo valor se encuentra entre un valor máximo y mínimo. El valor mínimo y máximo recibidos pueden estar o no estar almacenados en el diccionario. En caso de estar en el diccionario, ambos valores serán incluidos en la colección de retorno.

**Ejercicio 3:** Se desea almacenar un conjunto de tweets de diferentes temáticas de manera que se encuentren ordenados por el número de retweets más el número de veces que ha sido marcado como favorito dicho tweet. Para ello, se dispone de toda la información de cada tweet almacenada en un fichero en formato JSON[1], similar a XML pero más ligero y fácil de interpretar. Los tweets siempre seguirán la misma estructura: en primer lugar, tendrán un campo *statuses* que contiene $n$ elementos (uno por cada tweet recuperado). Dentro de cada tweet aparecerán diferentes valores, como el texto del tweet, el usuario que lo ha escrito, el número de retweets, etc. Se aconseja abrir el fichero JSON en la web http://www.jsoneditoronline.org/ para poder visualizar el contenido de cada tweet a modo de jerarquía.
El programa deberá mantener todos los tweets ordenados en función de su puntuación, que se calculará como el número de retweets más el número de veces que se ha marcado como favorito. Además, podrá consultar los tweets en un rango de puntuación, los mejores (cuya puntuación es superior a mejor*porcentaje, donde mejor es la puntuación del primer tweet) y los peores (cuya puntuación es inferior a mejor*porcentaje). La clase `TweetAnalysis` contiene los prototipos necesarios para la práctica, aunque se deja a elección del alumno la implementación de nuevos métodos que mejoren la legibilidad del código.
Para poder leer un fichero JSON desde Java utilizaremos la librería org.json[2], que nos permite abstraernos de parsear el fichero. A continuación se muestra un ejemplo para leer parte de un fichero de tweets:

[1] https://www.json.org/json-es.html
[2] https://search.maven.org/remotecontent?filepath=org/json/json/20171018/json-20171018.jar

```
String path = "darksouls.json";
BufferedReader bf = new BufferedReader(new FileReader(path));
String line = null;
String json = "";
while ((line = bf.readLine()) != null) {
	json += line;
}
bf.close();
JSONObject obj = new JSONObject(json);
JSONArray tweets = obj.getJSONArray("statuses");
System.out.println(tweets.length());
for (int i=0;i<tweets.length();i++) {
	JSONObject t = tweets.getJSONObject(i);
	System.out.println("Tweet "+i+": "+t.get("text"));
}
```