MINI-PROYECTO 3

Implementación de tablas con árboles binarios de búsqueda

Jordi Blanco Lopez

Jesus Fernandez Cid De Rivera

Curso: 2016/2017

Informe

En este proyecto se nos pide que implementemos las operaciones básicas sobre tablas usando árboles binarios de búsqueda.

Interfaz Map <K,V>

Interfaz que declara las operaciones que podremos realizar sobre una tabla:

```
public interface Map<K, V> {
    boolean isEmpty();
    void put(K key, V value);
    V get(K key);
    void remove(K key);
}
```

Las operaciones de la interfaz se comportan de la siguiente manera:

- *isEmpty():* devuelve true si la tabla está vacía y false, en caso contrario.
- *put(key, value):* asocia la clave key al valor value. Si key o value son null, lanzará NullPointerException.
- *get(key):* devuelve el valor asociado a la clave key o null, en caso de que la clave no tenga ningún valor asociado.
- *remove(key):* elimina la asociación que pudiera tener la clave key.

Clase BSTMap <K,V>

Esta clase que implementa la interfaz Map con un árbol binario de búsqueda. Como en la implementación es necesario que las claves sean comparables, la declaración de la clase es:

```
public class BSTMap<K extends Comparable<? super K>, V>
        implements Map<K, V> {

    ¿?
}
```

isEmpty()
Si la medida del árbol es 0 devuelve Cierto.

put(key, value)
Asocia la clave key al valor value. Si key o value son null, lanza NullPointerException.

Para hacer esto le pasamos la clave y el valor que queremos insertar y la raíz principal del árbol a un método privado que actualizara la raíz con los subárboles correspondientes.

Si se llega a una hoja crea un nuevo nodo con medida 1 y con la clave y el valor correspondientes.

Sino compara las claves, para ir actualizando el subárbol izquierdo/derecho y paso recursivamente al subárbol izquierdo/derecho para seguir por él, dependiendo de si actual es más pequeño que 0 (actualiza el izquierdo) o mas grande (actualiza el derecho). Si encontrara la clave actualizaría su valor.

Para finalizar actualizamos la medida del árbol.

get(key)
Devuelve el valor asociado a la clave key o null, en caso de que la clave no tenga ningún valor asociado.

Para hacer esto le pasamos la clave que queremos comparar y la raíz principal del árbol a un método privado.

Dicho método si recibe un árbol vacío devuelve null, sino compara la clave pasada por parámetro con la clave del árbol, si es mayor busca recursivamente en el árbol derecho, si es menor busca en el árbol izquierdo recursivamente, si es igual devuelve el valor asociado a la clave del nodo de ese árbol.

remove(key)
Elimina la asociación que pudiera tener la clave key.

Para hacer esto le pasamos la clave que queremos comparar y la raíz principal del árbol a un método privado.

Dicho método si recibe un árbol vacío devuelve null, sino compara las claves y eligiendo el subárbol según si es mayor o menos la clave y actualiza también. Si las claves son iguales, entonces si ese nodo no tiene hijo derecho, el hijo derecho lo cambiaremos directamente por el subárbol izquierdo y en caso contrario, cambiaremos el hijo izquierdo por el subárbol derecho.

En caso de que sea un nodo interno, Calculamos el mínimo del árbol derecho y luego borramos el mínimo y lo guardaremos como subárbol derecho.