AVLTREE

Es un tipo de Árbol Binario de Búsqueda equilibrado, el objetivo es asegurar que el árbol nunca se desequilibre de un valor de clave individual. Se basa en reequilibrados locales. La diferencia de altura entre el subárbol izquierdo y el derecho se le denomina factor de equilibrio, cada nodo tiene asociado uno; esta equilibrado cuando su factor de equilibrio es 0, en caso de no estar equilibrado se producirá rotaciones que pueden ser simples o dobles.

ADD:

Método público y booleano que le pasamos por parámetro un elemento del tipo parametrizado T y devuelve true o false. Este método nos permite insertar un nuevo elemento en el árbol.

Hace lo mismo que en el BSTree.

Busca el lugar donde se va a insertar el nuevo elemento, cumpliendo la propiedad de menores a la izquierda, mayores a la derecha (siempre insertando en hojas).

Si el elemento ya está, no se hace nada el método devuelve false. En caso contrario, se inserta en el lugar donde acaba la búsqueda devolviéndonos true. La búsqueda acaba sin éxito al llegar a un subárbol izquierdo o derecho que está vacío obtenemos un false como en el caso de estar ya el elemento.

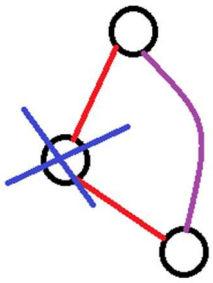
La única diferencia es que vuelve a calcular los nuevos factores de equilibrio y se comprueba que cumple la propiedad de equilibrio, en caso de no cumplirla se realizara una rotación que puede ser Simple (Izquierda o Derecha) o Doble (Izquierda-Derecha o Derecha-Izquierda).

REMOVE:

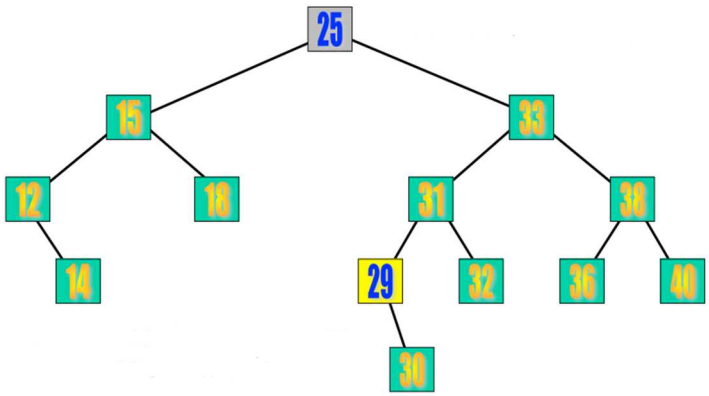
Método público y booleano que le pasamos por parámetro un elemento del tipo parametrizado T y devuelve true o false. Se hace lo mismo que en el BSTree.

Hay tres posibilidades a la hora de eliminar un elemento/nodo:

- Nodo sin hijos: Elimina el nodo sin ningún problema.
- Nodo con un solo hijo: El padre del nodo a eliminar pasa a apuntar al hijo del nodo que está siendo eliminado.



-
- Nodo con dos hijos: Reemplazamos el nodo a eliminar por el nodo mínimo de su subárbol derecho (o máximo de su subárbol izquierdo). Elimina el nodo mínimo (o máximo).
  Dicho nodo será una hoja o tendrá un solo hijo.



El 29 que es el menor de los mayores pasara a ser la raíz cuando eliminemos el 25.
Una vez eliminado el nodo se mira el factor de equilibrio del padre, si este tiene desequilibrio hay que hacer una rotación, mirando la rama contraria a la que borrado, una vez arreglado el desequilibrio debería mirar el nodo abuelo si está equilibrado no hay que hacer nada, pero si no lo está hay que volver a hacer una rotación, y miramos el bisabuelo, en cuanto uno de estos este equilibrado, paro de comprobar por encima de él, ya que no debería de haber desequilibrios por encima suya; es posible que las rotaciones se propaguen hasta la raíz.

CLEAR:

Método público que no devuelve nada. Este método elimina todo los elementos del árbol, por lo que obtenemos un árbol vacío después de ejecutar el método.

CONTAINS:

Método publico abstracto booleano, nos devuelve true si el árbol contiene el elemento que le especificamos (se lo pasamos como parámetro de tipo Objeto). Como tienen una propiedad de orden no es necesario buscar en todo el árbol.

ISEMPTY:

Método público abstracto booleano, nos devuelve true si el árbol no contiene elementos, es decir está vacío.

ITERATOR:

Método público y abstracto que nos devuelve un iterador sobre los elementos del árbol. Los elementos se devuelven en orden ascendente. Dentro de la clase BSTree hemos definido una clase que se llama TreeIterator, este hace un recorrido por el árbol INORDEN (Izquierda-Raiz-Derecha).

SIZE:

Método público y abstracto que nos devuelve el número de elementos que hay en el árbol.

FIND:

Método público de tipo T que le pasamos un elemento de tipo T que sirve como clave para localizar el elemento en el árbol. Busca el elemento especificado en el árbol y devuelve el valor del nodo, cuyo valor coincide con el elemento clave. Nos devuelve el valor del nodo que buscábamos o null en caso de que no lo encuentre.

-Ventajas de usar AVLTree frente a ArrayList:
Con ArrayList podemos acceder aleatoriamente en mayor medida a cualquier elemento, es decir, podemos acceder rápidamente. Pero las inserciones y eliminaciones con ArrayList requieren mover todos los elementos.

Las ventajas de usar AVLTree es que a la hora de buscar un elemento ya sea para añadir o eliminar podemos encontrarlo fácilmente, ya que están colocados de forma que los elementos de la izquierda de la raíz son menores que esta y los elementos de la derecha son mayores, por lo que no es necesario recorrer todo el árbol y está equilibrado, lo que facilita también mucho la tarea.