UNIVERSIDAD DE ALMERÍA

Estructuras de datos y Algoritmo I

Práctica 02: ejercicio 01

Curso: 2º Grado en Ingeniería Informática

Grupo docente: B1 **Grupo**: GTA3

Nombre y Apellidos:

Jefferson Max Tomalá Villarreal

Enunciado

Ejercicio 1. Árbol binario de búsqueda. Uso del árbol binario de búsqueda (BST, Binary Search Tree) para el control de una puerta de acceso a una red.

1) **Devolver el número de máquinas con un determinado valor de contador proporcionado como parámetro.**

```
Ejercicio 1: Devolver el número de maquinas
contador=1
(113.213.12.1, epicuro.ual.es)
(192.113.2.4, jupiter.ual.es)
(192.146.1.234, leo.ual.es)
contador=2
(192.146.1.234, voltaire.ual.es)
```

2) **Devolver el valor de contador del par (direcciónIP, máquina).**

```
Ejercicio 2: Devolver el valor del contador Par.
(192.146.1.234, voltaire.ual.es)
(192.146.1.234, voltaire.ual.es)

El valor del contador es: 2
```

3) **Se ha proporcionado todo lo relativo a la implementación del BSTree<T>, por lo que se pide que se EXPLIQUE en un documento (memoria) toda implementación suministrada, prestando especial atención a funciones principales de dicha estructura de datos como: add, remove, clear, contains, isEmpty, y al iterador iterator del tipo TreeIterator. Razone y justifique adecuadamente cada una de las ventajas e inconvenientes enumerados.**

Árboles binarios de búsqueda (Binary Search Trees):

Árbol binario de búsqueda es un árbol que tiene las siguientes propiedades:

- Cada nodo tiene un valor.
- Se define un orden total sobre esos valores.
- El subárbol izquierdo de un nodo contiene valores menores o iguales que el valor de dicho nodo.
- El subárbol derecho de un nodo contiene valores mayores o iguales que el valor de dicho nodo.

La ventaja más notable de los árboles binarios de búsqueda es que los algoritmos de ordenación y búsqueda relacionados como transversal inorden pueden ser muy eficientes.
Los árboles binarios de búsqueda son una estructura de datos fundamental usada para construir más estructuras de datos abstractas como conjuntos y arrays asociativos.

Add:

Asegura que esta colección contiene el elemento especificado. Devuelve true si la operación inserta un nuevo elemento en esta colección. Devuelve false si esta colección ya contiene el elemento especificado y no permite duplicados.
Parámetros: elemento de elemento que debe aparecer al menos una vez en esta colección.Returnos: true si se modifica esta colección

- La operación de inserción requiere, en el peor de los casos, un tiempo proporcional a la altura del árbol.
- La inserción es similar a la búsqueda y se puede dar una solución tanto iterativa como recursiva.

Remove:

Elimina una única instancia del elemento especificado de esta colección, si está presente; Devuelve true si se modifica esta colección.

La operación de borrado no es tan sencilla como las de búsqueda e inserción. Existen varios casos a tener en consideración:

- **Borrar un nodo sin hijos o nodo hoja**: simplemente se borra y se establece a nulo el apuntador de su padre.
- **Borrar un nodo con un subárbol hijo**: se borra el nodo y se asigna su subárbol hijo como subárbol de su padre.
- **Borrar un nodo con dos subárboles hijo**: la solución está en reemplazar el valor del nodo por el de su predecesor o por el de su sucesor en inorden y posteriormente borrar este nodo. Su predecesor en inorden será el nodo más a la derecha de su subárbol izquierdo (mayor nodo del subarbol izquierdo), y su sucesor el nodo más a la izquierda de su subárbol derecho (menor nodo del subarbol derecho).

Clear:

Este método elimina todos los elementos del árbol dejándolo vacio.

Contains:

Método que regresa un booleando en caso de que el elemento ítem pasado por parámetro exista/ este contenido dentro de la estructura de datos BSTree. True en caso de que exista, false en caso contrario.

- Cabe destacar que la búsqueda en este tipo de árboles es muy eficiente, representa una función logarítmica.

IsEmpty:

Método que consulta si el árbol está vacío, en caso de que lo esté devolverá un booleano “true” en caso que contenga elementos “false”.

Conclusión: Los tipos de árboles binarios de búsqueda para encontrar que tipo nos daría el mejor rendimiento para cada caso. Los montículos se encuentran como el tipo de árbol binario de búsqueda que mejor resultado promedio da, mientras que los árboles rojo-negro los que menor rendimiento medio nos aporta.