UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Estructuras de Datos

Práctica 12: Árbol Binario de Búsqueda

Profesora:
Amparo López Gaona
Ayudante: Adrián Aguilera Moreno
Ayudante de Laboratorio: Kevin Jair Torres Valencia

Objetivos

- Comprender e implementar la estructura de datos Árbol Binario de Búsqueda.
- Implementar dos tipos de recorrido: inorden y por niveles.

Introducción

Un Árbol Binario de Búsqueda (ABB) es una estructura de datos jerárquica que cumple con las siguientes propiedades:

- Cada nodo tiene cuando mucho dos hijos, denominados hijo izquierdo e hijo derecho.
- Todos los elementos en el subárbol izquierdo de cualquier nodo son menores o iguales que el valor de ese nodo.
- Todos los elementos en el subárbol derecho de cualquier nodo son mayores que el valor de ese nodo.
- Los subárboles izquierdo y derecho también son árboles binarios de búsqueda.

Estas propiedades permiten realizar búsquedas eficientes, ya que en cada comparación se descarta aproximadamente la mitad de los elementos restantes.

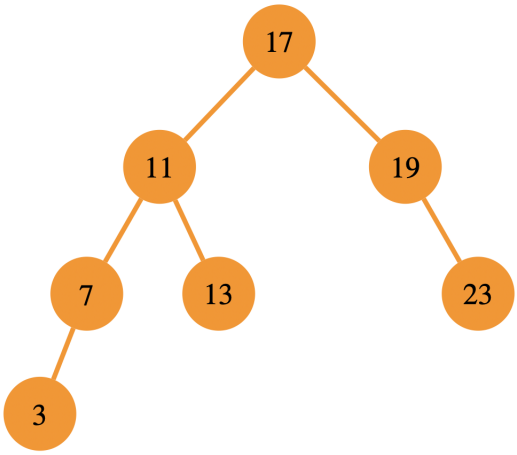

Figure 1: Representación de un Árbol Binario de Búsqueda

Operaciones básicas de un Árbol Binario de Búsqueda

Un Árbol Binario de Búsqueda soporta las siguientes operaciones principales:

- *Inserción*: Añade un nuevo elemento manteniendo las propiedades del ABB. Se empieza en la raíz y se desciende hasta encontrar el lugar adecuado para insertar el elemento, comparando con cada nodo en el camino.
- *Búsqueda*: Localiza un elemento en el árbol siguiendo el camino determinado por las comparaciones con los nodos. Si el elemento es igual al nodo actual, se ha encontrado; si es menor, se busca en el subárbol izquierdo; si es mayor, en el derecho.
- *Eliminación*: Remueve un elemento del árbol manteniendo las propiedades del ABB. Este es el caso más complejo y considera tres escenarios:
  - Nodo sin hijos: simplemente se elimina.
  - Nodo con un hijo: se reemplaza por su hijo.
  - Nodo con dos hijos: se reemplaza por su sucesor inorden (el elemento más pequeño del subárbol derecho) o su predecesor inorden (el elemento más grande del subárbol izquierdo).

Iterador

Recordemos que un `Iterator` proporciona una forma de acceder secuencialmente a los elementos de una colección sin exponer su estructura interna. Para nuestra implementación el iterador nos permite recorrer los nodos del árbol de diferentes maneras (inorden, por niveles) sin modificar la implementación del árbol.

- Inorden: Para lograr esto de manera iterativa (sin recursión), usamos una pila (`Stack`) que nos ayuda a "recordar" el camino mientras descendemos por el árbol. Para lograrlo, como inicialización se recorre el extremo izquierdo del árbol empujando los nodos a la pila hasta llegar a un nodo sin hijo izquierdo. Posteriormente por cada llamada a `next()`, se saca un nodo de la pila y se devuelve su valor. Y si el nodo tiene un hijo derecho, se empujan todos los nodos izquierdos de ese subárbol a la pila.
- Por nivel: Para realizar este recorrido recorrido usarán una cola `Queue()` que nos permite procesar los nodos en el orden en que fueron agregados. Para logralo, como inicialización se agrega la raíz a la cola. Posteriormente por cada llamada a `next()`, Se saca el nodo de la cola y se devuelve su valor. Y se agregan sus hijos izquierdo y derecho (si existen) a la cola.

Nodo

La clase Nodo representa los nodos del árbol. Cada nodo almacena

- Un dato de tipo `T`
- Una referencia al hijo izquierdo.
- Una referencia al hijo derecho.

Desarrollo

En esta práctica, deberán completar la implementación de un Árbol Binario de Búsqueda genérico, siguiendo el principio de separación de interfaz e implementación. Se te proporcionarán algunos archivos base que deberás extender.

Para ello tendrán que implementar los métodos en las siguientes clases:

- ArbolBB:
  - Debe implementar todos los métodos de la interfaz `ArbolBinarioBusqueda`, es decir, las operaciones de inserción, búsqueda y eliminación.
  - Para el caso de la eliminación, considerar los tres casos (nodo sin hijos, con un hijo, con dos hijos).
- IteradorInorden:
  - Deben usar una pila para implementar el recorrido inorden.
  - Implementar los métodos de `hasNext` y `next`, de acuerdo a lo anterior.
- IteradorNivel
  - Deben usar una cola para implementar el recorrido por niveles.
  - Implementar los métodos de `hasNext` y `next`, de acuerdo a lo anterior.

Archivos proporcionados

- ArbolBinarioBusqueda: Interfaz que define el comportamiento de un Árbol Binario de Búsqueda.
- Nodo: Representa un nodo del Árbol Binario de Búsqueda.
- ArbolBB: Implementación de la interfaz de un Árbol Binario de Búsqueda.
- Iterador: Interfaz para el Iterador
- IteradorInorden: Implementación de un iterador para recorrer un Árbol Binario de Búsqueda en orden (inorden).
- IteradorNivel: Implementación de un iterador para recorrer un Árbol Binario de Búsqueda por niveles.
- Main: Clase para probar la implementación de la estructura.

Formato de Entrega

1. Las prácticas se entregarán en parejas.
2. NO incluir los archivos .class dentro de la carpeta.
3. Los archivos de código fuente deben estar documentados.
4. Se pueden discutir y resolver dudas entre los integrantes del grupo. Pero cualquier práctica plagiada total o parcialmente será penalizada con cero para los involucrados.
5. La práctica se debe subir al Github Classroom correspondiente.
6. La entrega en classroom debe contener el link HTTPS y SSH de su repositorio y es lo único que se debe entregar.
7. El horario y día de entrega se acordará en la clase de laboratorio y no deberá sobrepasar 2 clases de laboratorio.