UNIVERSIDAD AUTONOMA GABRIEL RENE MORENO

FACULTAD DE INGENIERIA Y CIENCIAS DE LA COMPUTACION Y TELECOMUNICACIONES

ESTRUCTURA DE DATOS II

CONTENIDO:

TAREA-2. LISTA DE ÁRBOLES.

PORCENTAJE TERMINADO: 100%

GRUPO: 14

INTEGRANTES	DT	HG	HI	EVAL
Flores Veizaga Eudenia Gandira	1	1	1	100
Garcia Taboada Brayan Albaro	1	1	1	100
Gedalge Cayhuara Cristian Gabriel	1	1	1	100
Haquin Serrano Rodrigo	1	1	1	100
Hernandez Lijeron Roly	1	1	1	100

//DT, días trabajados

//HG, horas grupo

//HI, horas individual

// Eval

Fecha de presentación : jueves, 09 de mayo de 2024

Fecha Presentada : jueves, 09 de mayo de 2024

Días de Atraso : 0

```
import java.util.ArrayList;
public class ListaArbol {

   private int cantElem;
   private int max;
   private Arbol arbol[];

   public ListaArbol() {
      this.max = this.cantElem = 10;
      arbol = new Arbol[10];
      for (int i = 0; i < 10; i++) {
         arbol[i] = new Arbol();
      }
   }

   public void generar(int n, int a, int b) {
      for (int i = 1; i <= n; i++) {
         int r = (int) (a + (b - a) * Math.random());
         int lastDigit = r % 10;
         arbol[lastDigit].insertar(r);
      }
   }

   public void mostrar() {
      for (int i = 0; i < 10; i++) {
         arbol[i].inOrden();
         System.out.println("");
      }
   }

   public ArrayList<Integer> obtenerNumerosPorDigito(int digito) {
```

```
        if (digito < 0 || digito > 9) {
            return new ArrayList<>();
        }
        return arbol[digito].obtenerNumeros();
    }

    // Consultas adicionales
    public int cantidadTotalDeNumeros() {
        int total = 0;
        for (Arbol a : arbol) {
            total += a.cantidadDeNodos();
        }
        return total;
    }

    public int cantidadDeNumerosPorDigito(int digito) {
        if (digito < 0 || digito > 9) {
            return 0;
        }
        return arbol[digito].cantidadDeNodos();
    }

    public ArrayList<Integer> numerosConMasDeNDigitos(int n) {
        ArrayList<Integer> result = new ArrayList<>();
        for (Arbol a : arbol) {
            result.addAll(a.numerosConMasDeNDigitos(n));
        }
        return result;
    }

}
```

```
//--------------------------------------
import java.util.ArrayList;

class Arbol {

   private Nodo raiz;

   public Arbol() {
      raiz = null;
   }

   public void insertar(int x) {
      raiz = insertar(x, raiz);
   }

   private Nodo insertar(int x, Nodo p) {
      if (p == null) {
         return new Nodo(x);
      }
      if (x < p.elem) {
         p.izq = insertar(x, p.izq);
      } else {
         p.der = insertar(x, p.der);
      }
      return p;
   }

   public void inOrden() {
      inOrden(raiz);
   }
```

```
private void inOrden(Nodo p) {
   if (p == null) {
      return;
   }
   inOrden(p.izq);
   System.out.println(p.elem);
   inOrden(p.der);
}

public ArrayList<Integer> obtenerNumeros() {
   ArrayList<Integer> numeros = new ArrayList<>();
   obtenerNumeros(raiz, numeros);
   return numeros;
}

private void obtenerNumeros(Nodo p, ArrayList<Integer> numeros) {
   if (p == null) {
      return;
   }
   obtenerNumeros(p.izq, numeros);
   numeros.add(p.elem);
   obtenerNumeros(p.der, numeros);
}

public int cantidadDeNodos() {
   return contarNodos(raiz);
}

private int contarNodos(Nodo p) {
   if (p == null) {
```

```
            return 0;
        }
        return 1 + contarNodos(p.izq) + contarNodos(p.der);
    }

    public ArrayList<Integer> numerosConMasDeNDigitos(int n) {
        ArrayList<Integer> result = new ArrayList<>();
        encontrarNumerosConMasDeNDigitos(raiz, n, result);
        return result;
    }

    private void encontrarNumerosConMasDeNDigitos(Nodo p, int n, ArrayList<Integer>
result) {
        if (p == null) {
            return;
        }
        encontrarNumerosConMasDeNDigitos(p.izq, n, result);
        if (String.valueOf(p.elem).length() > n) {
            result.add(p.elem);
        }
        encontrarNumerosConMasDeNDigitos(p.der, n, result);
    }
}
```

La diferencia principal entre una lista basada en arrays y una basada en árboles radica en sus tiempos de ejecución en diferentes operaciones.

Inserción:

En una lista basada en arrays, la inserción al final de la lista tiene un tiempo de ejecución promedio de O(1) si hay espacio disponible en el array subyacente. Sin embargo, si el array está lleno y necesita redimensionarse, el tiempo de ejecución podría ser O(n), donde n es el tamaño actual del array.

En un árbol, la inserción tiene un tiempo de ejecución promedio de O(log n) en un árbol binario de búsqueda balanceado. Esto se debe a que cada inserción divide el espacio de búsqueda a la mitad. Sin embargo, si el árbol no está balanceado, la inserción podría tener un peor tiempo de ejecución de O(n) en el peor caso, donde n es la altura del árbol.

Búsqueda:

En una lista basada en arrays, la búsqueda lineal tiene un tiempo de ejecución promedio de O(n), donde n es el tamaño de la lista.

En un árbol, la búsqueda tiene un tiempo de ejecución promedio de O(log n) en un árbol binario de búsqueda balanceado. En el peor caso, el tiempo de ejecución de la búsqueda puede ser O(n) si el árbol no está balanceado y se asemeja a una lista enlazada.

Eliminación:

En una lista basada en arrays, eliminar un elemento en una posición específica tiene un tiempo de ejecución promedio de O(n) debido a la necesidad de desplazar elementos.

En un árbol, la eliminación tiene un tiempo de ejecución promedio de O(log n) en un árbol binario de búsqueda balanceado. Sin embargo, en el peor caso, la eliminación puede tener un tiempo de ejecución de O(n) si el árbol no está balanceado y se asemeja a una lista enlazada.

En general, los árboles son más eficientes en operaciones de búsqueda y eliminación en comparación con las listas basadas en arrays. Sin embargo, la implementación de árboles puede requerir más espacio en memoria debido a la estructura de nodos enlazados. La elección entre una lista basada en arrays y un árbol depende de los requisitos específicos del problema y las operaciones que se realizarán con mayor frecuencia.