TECNOLOGICO NACIONAL DE MEXICO

INSTITUTO TECNOLOGICO DE ORIZABA

Asignatura: Estructura de Datos

Carrera: Ingeniería en Informática

Estudiante:

- Dorantes Rodríguez Diego Yael - No. Control 21010184

Profesora: María Jacinta Martínez Castillo

Grupo: 3a3A

Fecha de entrega: 2/06/2023

Introduccion

En este informe de la Unidad 4, vamos a explorar y explicar los códigos y programas relacionados con los conceptos de Datos Simples o arreglos. Además, se analizarán ejemplos prácticos sobre pilas, colas y listas enlazadas, incluyendo las listas doblemente ligadas.

El propósito principal de este informe es documentar y resumir todo lo que hemos realizado en las diferentes unidades del curso de estructura de datos. A través de un enfoque paso a paso, se mostrará cómo hemos progresado en los diferentes temas y cómo hemos aplicado los códigos proporcionados por la maestra.

Este informe también nos brinda la oportunidad de repasar y reforzar nuestros conocimientos prácticos adquiridos en las unidades anteriores. Seremos capaces de resolver problemas por nuestra cuenta, basándonos en apuntes y materiales previos, y así verificar si nuestros resultados cumplen con las expectativas. Este proceso de repaso será beneficioso tanto para cada unidad individual como para el conjunto del curso de estructura de datos en este semestre.

En cuanto al proyecto académico en sí, se centra en el estudio y la comprensión de cómo trabajar con nodos en listas simplemente enlazadas. Estas listas nos permiten insertar y eliminar elementos en cualquier posición: al principio, en el medio o al final. Sin embargo, en ciertos escenarios de programación, es necesario restringir estas operaciones de inserción y eliminación solo al principio o al final. Por lo tanto, se presentarán las pilas y las colas como dos estructuras de datos útiles en dichas situaciones.

Competencia especifica

Conoce y comprende las diferentes estructuras de datos, su clasificación y forma de manipularlas para buscar la manera más eficiente de resolver problemas.

Genéricas:

- Utilizar las clases predefinidas para el manejo de pilas, colas y listas enlazadas (dinámicas) y describir en un texto la diferencia de hacerlo con arreglos.
- Utilizar las estructuras lineales en la elaboración de códigos para la resolución de problemas elaborando un reporte.

Marco teorico

Listas simples enlazadas

Las listas simplemente enlazadas son una estructura de datos lineal en la que los elementos, llamados nodos, están conectados mediante enlaces o punteros unidireccionales. Cada nodo contiene un dato y un puntero que apunta al siguiente nodo en la lista.

La característica principal de las listas simplemente enlazadas es que permiten la inserción y eliminación eficiente de elementos en cualquier posición de la lista, ya sea al principio, en el medio o al final. Esto se logra ajustando los punteros de los nodos vecinos para redirigir el flujo de la lista.

La ventaja de las listas simplemente enlazadas radica en su flexibilidad para agregar o quitar elementos en cualquier lugar sin tener que desplazar todos los elementos posteriores, como sucede en otros tipos de estructuras de datos como los arreglos. Esto las hace especialmente útiles en situaciones donde se requiere una operación frecuente de inserción o eliminación de elementos en diferentes posiciones.

Sin embargo, una limitación de las listas simplemente enlazadas es que no se puede acceder directamente a un elemento en una posición específica mediante un índice, como se hace en los arreglos. Para acceder a un elemento en una lista simplemente enlazada, se debe recorrer la lista desde el principio hasta la posición deseada, siguiendo los punteros de los nodos.

Lista simplemente enlazada.

Nodos

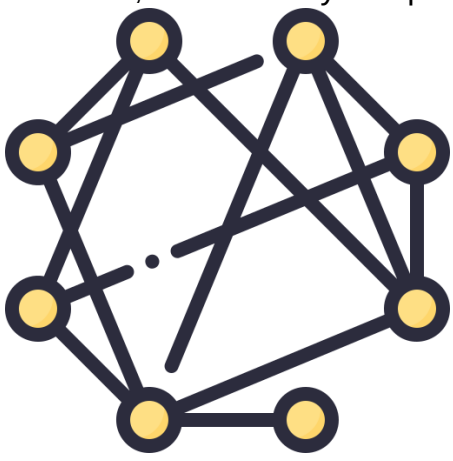
Los nodos son elementos fundamentales en las estructuras de datos, como las listas simplemente enlazadas. Cada nodo contiene dos componentes principales: un dato o valor y uno o más punteros.

El dato o valor es la información que se almacena en el nodo. Puede ser cualquier tipo de dato, como un número, una cadena de texto o incluso un objeto más complejo. El dato representa la información que queremos almacenar en la estructura de datos.

Los punteros son referencias a otros nodos en la estructura. En el caso de las listas simplemente enlazadas, un nodo tiene al menos un puntero que apunta al siguiente nodo en la lista. Este puntero se conoce como "siguiente" o "next". A través de los punteros, los nodos están interconectados y forman la estructura lineal de la lista.

Cuando se crea una lista simplemente enlazada, se utiliza un puntero especial llamado "cabeza" o "head" para indicar el primer nodo de la lista. A partir de este nodo inicial, se pueden seguir los punteros "next" para recorrer todos los nodos de la lista secuencialmente.

En cada nodo, se almacena tanto el dato como el puntero al siguiente nodo. Esto permite que los nodos estén enlazados de manera secuencial, lo que facilita la inserción, eliminación y búsqueda de elementos en la lista.



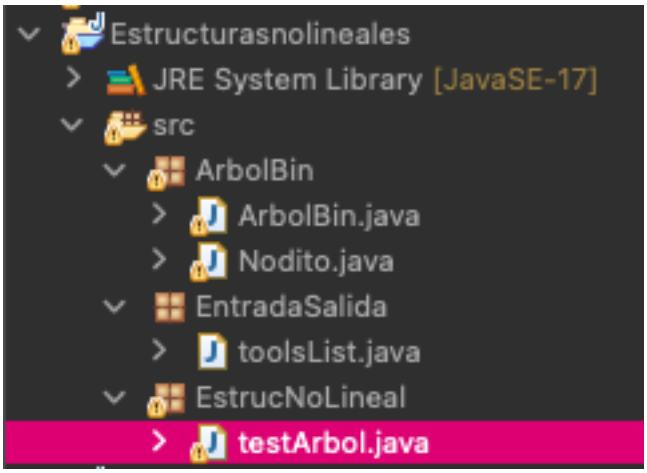
Funciones

Las listas simples enlazadas ofrecen una serie de funciones principales o elementales para manipular y trabajar con los elementos de la lista. Algunas de estas funciones incluyen:

1. Inserción al principio: Permite agregar un nuevo elemento al inicio de la lista. Para ello, se crea un nuevo nodo con el dato deseado y se ajustan los punteros para que el nuevo nodo sea el primero de la lista.
2. Eliminación en una posición específica: Permite eliminar un elemento en una posición determinada de la lista. Se recorre la lista hasta llegar a la posición deseada, se ajustan los punteros para saltar el nodo a eliminar y se libera la memoria ocupada por dicho nodo.
3. Búsqueda: Permite buscar un elemento en la lista. Se recorre la lista comparando cada nodo con el elemento buscado hasta encontrar una coincidencia o llegar al final de la lista.

Desarrollo de la practica

Creamos el proyecto como EstructurasnoLineales en donde comezamos a crear y trabjar en los paquetes de arbolbin donde se alojara la calse ArbolBin y Nodito, creamos el paquete EntradaySalida donde trasladamos las Tools que hemos utilizado en todo el semestre, por ultimo creamos el paquete de EstreucturaNoLineal donde se alojara la clase de tesArbol que es donde estara nuestro menu



Iniciamos en la clase de ArbolBin que es donde estaran nuestros metodos principales del proyecto

```java

public class ArbolBin<T> {

    private Nodito raiz;

    public ArbolBin(){
        raiz = null;
    }

    public Nodito getRaiz() {
        return raiz;
    }

    public void setRaiz(Nodito raiz) {
        this.raiz = raiz;
    }

    public boolean arbolVacio(){
        return raiz==null;
    }

    public void vaciarArbol(){
        raiz=null;
    }

    public void insertarArbol(T info){
        Nodito p = new Nodito(info);
        if(arbolVacio())
            raiz = p;
        else{
            Nodito padre = buscarPadre(raiz,p);
            if ((int) p.info >= (int) padre.info)
                padre.setDer(p);
            else
                padre.setIzq(p);
        }
    }

    public Nodito buscarPadre(Nodito actual,Nodito p){
        Nodito padre = null;
        while(actual!=null){
            padre = actual;
            if((int)p.info>=(int)padre.info)
                actual = padre.getDer();
            else
                actual = padre.getIzq();
        }
        return padre;
    }

```

```java
public String preorden(Nodito r){
    if(r!=null){
        return r.info + " - " + preorden(r.izq) + " - " + preorden(r.der);
    }
    else
        return " ";
}

public String posorden (Nodito r) {
     if (r!=null) {
         return posorden(r.izq)+" - "+posorden(r.der)+" - "+r.info;

     }
     else return " ";
 }


 public String inorden(Nodito r) {
     if (r!=null) {
         return inorden(r.izq)+" - "+r.info+" - "+inorden(r.der);

     }
     else return " ";
 }



 public String enorden(Nodito r) {
     if (r!=null) {
         return enorden(r.der)+" - "+r.info+" - "+enorden(r.izq);

     }
     else return " ";
 }

 public Nodito buscarDato(Nodito r, int dato) {
     while(r!=null) {
         if(r.getInfo().equals(dato)) {
             return r;
         } else {
             int i =(int)r.info;
             if(dato>i) {
                 r=r.getDer();
             }else {
                 r=r.getIzq();
             }
         }
     }
     return r;
 }
```

```java
public String imprimirHojas(Nodito nodo) {
        StringBuilder cad = new StringBuilder();
        imprimirHojasRecursivo(nodo, cad);
        return cad.toString();
    }

private void imprimirHojasRecursivo(Nodito nodo, StringBuilder cad) {
        if (nodo == null) {
            return;
        }
        if (nodo.izq == null && nodo.der == null) {
            cad.append(nodo.info).append(" ");
        }
        imprimirHojasRecursivo(nodo.izq, cad);
        imprimirHojasRecursivo(nodo.der, cad);
    }

public String imprimirNodInter(Nodito nodo) {
        String cad = "";
        if (nodo != null) {
            if (nodo.izq != null || nodo.der != null) {
                if(nodo.info!=raiz.info)
                cad = nodo.info + " ";
            }
            return cad + imprimirNodInter(nodo.izq) + imprimirNodInter(nodo.der);
        } else {
            return "";
        }
    }

public int obtenerAltura(Nodito nodo) {
        if (nodo == null) {
            return 0;
        } else {
            int alturaIzq = obtenerAltura(nodo.izq);
            int alturaDer = obtenerAltura(nodo.der);
            return Math.max(alturaIzq, alturaDer) + 1;
        }
    }
```

```java
public static void Recursivo(Nodito node, int level) {
        if (node == null) {
            return;
        }

        Recursivo(node.der, level + 1);

        StringBuilder indentation = new StringBuilder();
        for (int i = 0; i < level; i++) {
            indentation.append("    ");
        }
        String prefix = (level == 0) ? "" : indentation.toString() + ">";
        System.out.println(prefix + node.info);
      Recursivo(node.izq, level + 1);
    }

public void imprimirArbol(Nodito nodo, String prefijo, boolean esUltimo) {
        if (nodo == null) {
            return;
        }
        String separador = esUltimo ? ">" : ">";
        String nuevoPrefijo = prefijo + (esUltimo ? ">    " : ">   ");
        imprimirArbol(nodo.der, nuevoPrefijo, false);
        System.out.println(prefijo + separador + nodo.info);
        nuevoPrefijo = prefijo + (esUltimo ? ">   " : ">    ");
        imprimirArbol(nodo.izq, nuevoPrefijo, true);
    }
```

En este otro paquete se encuentran los metodos pertenecientes a la clase Nodito bb

```java
package ArbolBin;


public class Nodito<T>{
    public T info;
    public Nodito izq;
    public Nodito der;

    public Nodito(T info){
        this.info = info;
        this.izq = null;
        this.der = null;
    }

    public T getInfo(){
        return info;
    }

    public void setInfo(T info) {
        this.info = info;
    }

    public Nodito getIzq() {
        return izq;
    }

    public void setIzq(Nodito izq) {
        this.izq = izq;
    }

    public Nodito getDer() {
        return der;
    }

    public void setDer(Nodito der) {
        this.der = der;
    }
}
```

Ahora en este paquete que es nuestra clase testArbol es donde tendremos nuestro menu por el cual hare llamado a nuestro metodos mediantes los CASE de insertar, Recorridos, Buscar, Hojas, Altura y Ver.

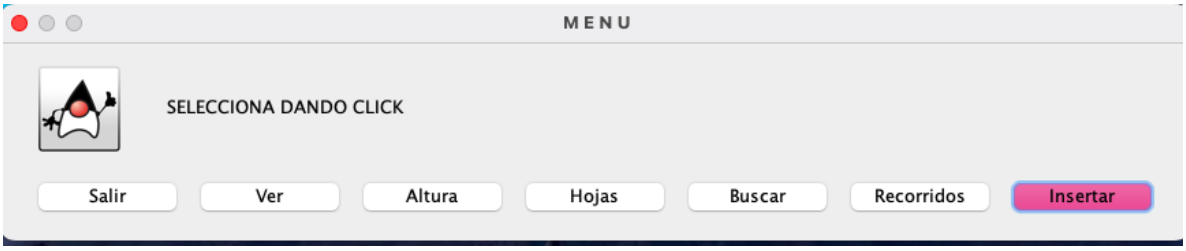
```java
package EstrucNoLineal;

import EntradaSalida.toolsList;

public class testArbol {
    public static void main(String[] args) {
        String menu = "Insertar,Recorridos,Buscar,Hojas,Altura,Ver,Salir";
        menu3(menu);

    }

    public static String boton(String menu) {
        String valores[]=menu.split(",");
        int n;
        n = JOptionPane.showOptionDialog(null," SELECCIONA DANDO CLICK ", " M E N U",
                JOptionPane.NO_OPTION,
                JOptionPane.QUESTION_MESSAGE,null,
                valores,valores[0]);
        return ( valores[n]);
    }
    public static void menu3(String menu)
    {

        ArbolBin <Integer> arbol;
        arbol = new ArbolBin();

        String sel="";
        do {
            sel=boton(menu);
            switch(sel){
                case"Insertar":
                    arbol.insertarArbol(toolsList.leerInt("Dato"));
                    break;
                case"Recorridos":
                    toolsList.imprimePantalla(arbol.preorden(arbol.getRaiz())+"\n"+
                            "pos"+arbol.posorden(arbol.getRaiz())+"\n"+
                            "en"+arbol.enorden(arbol.getRaiz())+"\n"+
                            "in"+arbol.inorden(arbol.getRaiz()));
                    break;
```

```
case"Buscar":
    if (arbol.arbolVacio()) {
        toolsList.imprimeErrorMsg("Arbol vacio");
    } else {
        int datoBuscar = toolsList.leerInt("Ingresa valor a buscar");
        Nodito resultado = arbol.buscarDato(arbol.getRaiz(), datoBuscar);

        if (resultado != null) {
            toolsList.imprimePantalla("El dato " + datoBuscar + " se encuentra en el arbol");
        } else {
            toolsList.imprimePantalla("El dato " + datoBuscar + " no se encuentra en el Árbol");
        }
    }
    break;

case"Hojas":
    if (arbol.arbolVacio()) {
        toolsList.imprimeErrorMsg("Arbol vacio");
    } else {
        toolsList.imprimePantalla("Las hojas del arbol son: \n" + arbol.imprimirHojas(arbol.getRaiz()));
        toolsList.imprimePantalla("Los interiores del arbol son: \n" + arbol.imprimirNodInter(arbol.getRaiz()));

    }
    break;

case"Altura":
    if (arbol.arbolVacio()) {
        toolsList.imprimeErrorMsg("Arbol vacio");
    } else {
        toolsList.imprimePantalla("La altura del arbol es: \n" + arbol.obtenerAltura(arbol.getRaiz()));
    }
    break;
```

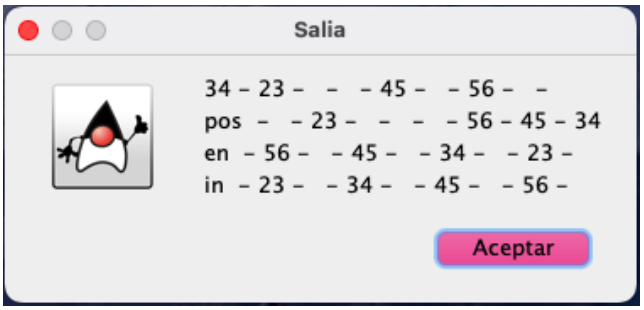
```
                case"Ver":
                    if (arbol.arbolVacio()) {
                        toolsList.imprimeErrorMsg("Arbol vacio");
                    } else {
                        System.out.println("La estructura del arbol es la siguiente: \n");
                        arbol.Recursivo(arbol.getRaiz(), 0);
                    }
                    break;

                case "Salir":;  break;
            }
        }while(!sel.equalsIgnoreCase("Salir"));
    }
}
```

Resultados:

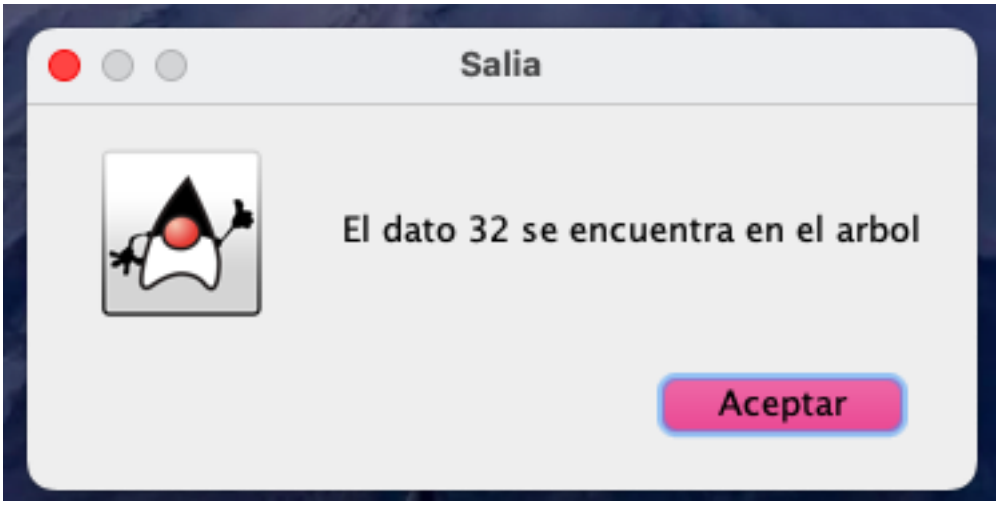
Se comenzara a mostrar el menu con los metodos usados el cual contiene,Insertar, Recorridos, Buscar, Hojas, Altura, Ver y Salir.



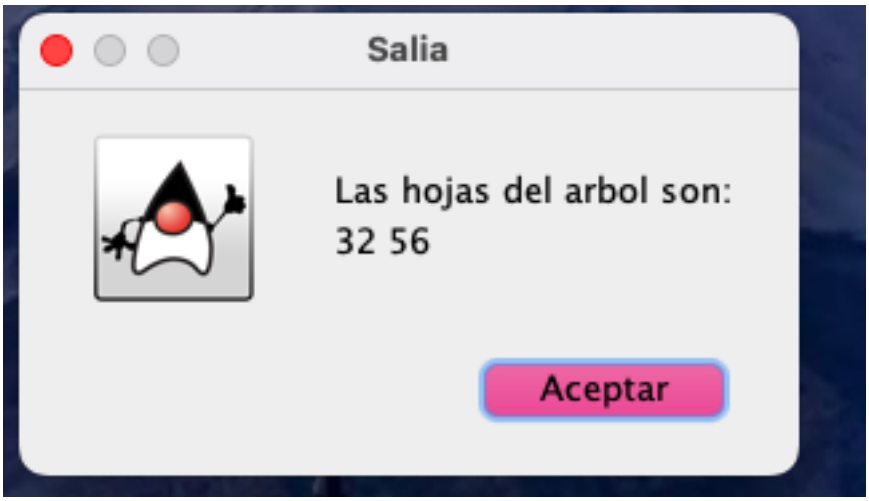
Una ves insertados los valores seleccionamos la opcion de recorridos y nos aparecera asi



Ahora hemos seleccionados la opcion de Buscar e ingresamos el elemento 32 donde nos dira que se encuentra en el arbol

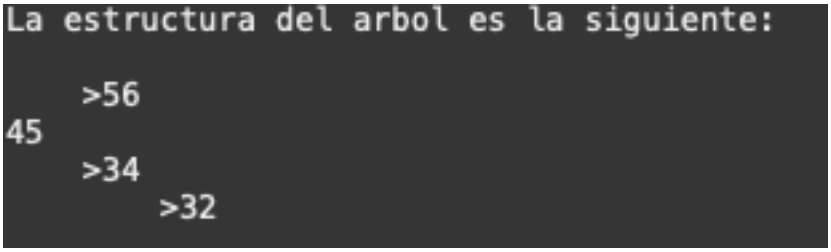


Elejimos la opcion de hojas y nos arroja que las hojas del arbol son las 32 y 56 ademas de mostrarnos el nodo interior que en este caos es el 34



La altra del arbol se identifica como en 3

Acontinuacion para poder vizualizar el arbol se muetra de la siguiente manera



Conclusion

En conclusión, las listas simples enlazadas, junto con las colas y las pilas, son estructuras de datos fundamentales en programación. Cada una de estas estructuras tiene sus propias características y funciones principales.

Las listas simples enlazadas proporcionan flexibilidad en la inserción y eliminación de elementos en cualquier posición de la lista. Esto las hace especialmente útiles cuando se requiere manipular la lista en diferentes partes y no se necesita acceder a los elementos de manera directa mediante un índice. Sin embargo, el acceso a un elemento en una posición específica puede requerir recorrer la lista, lo que puede aumentar la complejidad temporal.

Por otro lado, las colas y las pilas son estructuras de datos especializadas que restringen las operaciones de inserción y eliminación a un extremo de la estructura. Las colas siguen el principio "FIFO" (First-In-First-Out), lo que significa que el primer elemento en ser insertado es el primero en ser eliminado. Por otro lado, las pilas siguen el principio "LIFO" (Last-In-First-Out), donde el último elemento en ser insertado es el primero en ser eliminado. Estas estructuras son eficientes para implementar ciertos algoritmos y resuelven problemas específicos de manera eficiente.

Bibliografia

https://www.it.uc3m.es/java/2011-12/units/pilas-colas/guides/2/guide_es_solution.html

https://analisisyprogramacionoop.blogspot.com/2017/07/lista-simplemente-enlazada-C-sharp.html

https://www.uv.mx/personal/ermeneses/files/2021/08/Clase8-Arboles.pdf

http://www.itnuevolaredo.edu.mx/takeyas/Apuntes/Estructura%20de%20Datos/Apuntes/07-ABB.pdf

https://alaisecure.co/glosario/nodos-que-son/

https://definicion.de/nodo/