

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

PROGRAMACIÓN III

ISWZ2102 / 5540 / 2025

CRISTIAN LÓPEZ

Taller Semana 13: Árboles

Implementación de un Árbol Cuaternario

DOCUMENTACIÓN

Resumen modificaciones realizadas:

Basicamente al proyecto inicial del arbol, modifique;

- **Hice reemplazo de referencias:**
Se reemplazaron las referencias a hijos binarios (Nodo izquierda, Nodo derecha) por cuatro nuevas referencias para hijos cuaternarios: Nodo hijo1, Nodo hijo2, Nodo hijo3, Nodo hijo4.
- **Nueva Característica:**
Enumeración PosicionHijo: Añadi una enumeración (public enum PosicionHijo { HIJO1, HIJO2, HIJO3, HIJO4 }) para especificar de manera clara la posición de un nuevo hijo al insertarlo.
- **Modificación para la Inserción de Nodos (anadirNodo):**

Firma del Método: Cambió de `anadirNodo(Nodo nodo, Nodo padre, boolean esLzq)` a `anadirNodo(Nodo nuevoNodo, Nodo padre, PosicionHijo posicion)`

Lógica Interna: Se reemplazó la lógica if-else de `esLzq` por una estructura switch que asigna el `nuevoNodo` a `padre.hijo1`, `padre.hijo2`, `padre.hijo3`, o `padre.hijo4` según el valor de `posicion`.

Validación: Se mantuvo y extendió la validación para asegurar que no se sobrescriba un hijo si la posición ya está ocupada.

- **Modificación de Eliminación de Nodos (`eliminarNodo`):**

Nuevo Método: Se implementó `public boolean eliminarNodo(String etiquetaNodo)` para desconectar un nodo de su padre y eliminar recursivamente todos los nodos de su subárbol de la lista `nodos`.

Métodos Auxiliares: Se añadieron `private Nodo encontrarPadre(Nodo actual, String etiquetaHijo)` para localizar el padre de un nodo, y `private void eliminarSubarbolDeLista(Nodo nodo)` para gestionar la eliminación de los nodos de la lista `nodos`.

Comentarios: Documentan la naturaleza de la eliminación (poda de subárbol, sin reestructuración compleja).

- **Adaptación de Recorridos (`bfs`, `dfs`, `preorden`, `inorden`, `postorden`):**

Lógica Interna: Todas las implementaciones de los recorridos (tanto iterativos como recursivos) se modificaron para visitar o procesar los cuatro posibles hijos (`hijo1`, `hijo2`, `hijo3`, `hijo4`) en un orden definido y consistente.

Comentario inorden: Se añadió una nota explicando que el término "inorden" se ha adaptado, ya que su significado estricto es para árboles binarios; aquí implica un orden específico de visita entre el primer hijo, la raíz y los demás hijos.

- **Adaptación de Matriz de Adyacencia (getMatrizAdyacencia):**

Lógica Interna: Se extendió el bucle para incluir y representar las conexiones a hijo3 y hijo4 en la matriz.

- **Comentarios Añadidos:**

Documentación detallada sobre cada método y las decisiones de diseño para el árbol cuaternario.

- **Adaptación del Dibujo (dibujarArbol):**

Lógica de Posicionamiento: La parte más crítica. Se modificaron los cálculos de las coordenadas childX para distribuir horizontalmente los cuatro hijos. Se ajustaron los multiplicadores (e.g., * 1.5, * 0.5) para intentar una distribución visualmente equilibrada.

Reducción de dimensionX: La dimensionX se sigue reduciendo a la mitad ($\text{dimensionX} / 2$) en cada nivel para asegurar que el árbol se comprima a medida que profundiza.

Calidad de Renderizado: Se añadieron `g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);` y `g2d.setStroke(new BasicStroke(1.5f));` para mejorar la apariencia de los nodos y líneas.

- **Corrección de Errores:**

Error incompatible types: possible lossy

conversion from double to int: Se corrigió el casting explícito a int ((int)(horizontalSpacing * 1.5)) en los cálculos de childX para evitar la pérdida de precisión y el error de compilación.

- **Comentarios Añadidos:**

Detalles sobre cómo se calculan las posiciones para los cuatro hijos y la importancia de los ajustes de espaciado.

Modificaciones en la Interfaz de Usuario (Formulario ArbolGUI.form necesario):

cbIzqDer a cbPosicionHijo: Se instruyó para cambiar el nombre y el tipo del JComboBox en el diseñador de UI para que muestre las opciones de Arbol.PosicionHijo.

Nuevos Controles para Eliminación: Se añadió la necesidad de incluir un JButton (btnEliminarNodo) y un JTextField (txtNodoAEliminar) en el formulario para la nueva funcionalidad de eliminación.

- **Integración del Dibujo en panelArbol:**

Línea Crucial: Se añadió panelArbol.add(arbolGrafico, BorderLayout.CENTER); en el constructor de ArbolGUI.

Razón: Para asegurar que la instancia de ArbolGrafico (que es un JPanel de dibujo) sea efectivamente añadida al panelArbol en la GUI, permitiendo que se visualice el árbol. Se confirmó que panelArbol ya tiene BorderLayout en el diseñador de UI, por lo que no fue necesario setLayout en Java.

- **Modificación de btnAgregarNodo:**
 - **Lógica de Inserción:** Ahora obtiene la PosicionHijo seleccionada del cbPosicionHijo y la pasa al método arbol.anadirNodo().
 - Manejo de Errores:** Se mejoraron los mensajes de error para el usuario si el nodo padre no se encuentra o si se intenta añadir una raíz cuando ya existe.
 - Corrección de Error (nodos has private access):** Se eliminó la línea `int tempNumNodos = arbol.nodos.size();` que intentaba acceder directamente a un miembro private, resolviendo el error de compilación. Se explicó que la implicación es que una etiqueta generada se "pierde" si la adición falla, pero esto es aceptable por simplicidad.
- **Nueva Funcionalidad (btnEliminarNodo):**
 - Listener Implementado:** Se añadió un ActionListener para el nuevo botón btnEliminarNodo que obtiene la etiqueta del nodo a eliminar del txtNodoAEliminar, llama a `arbol.eliminarNodo()`, y actualiza la GUI (lista de nodos y el dibujo).
 - Confirmación:** Incluye un cuadro de diálogo de confirmación antes de la eliminación.
- **Actualización de Dibujo (repaint):**
 - Consistencia:** Se cambió el uso directo de `getGraphics()` por `panelArbol.repaint()` en btnAgregarNodo y btnDibujarArbol para seguir las mejores prácticas de Swing en el manejo del redibujado.
- **Comentarios Añadidos:**

Documentación de la lógica de cada ActionListener y la integración con las nuevas funcionalidades del Arbol.

Explicación de técnicas y algoritmos para el árbol cuaternario

Resumen de lo que investigue, junto a conceptos y y algoritmos que implemente para insercion de lo nodo, su elimancion y recusiones:

- **Inserción de Nodos:**

Algoritmo: Si no hay raíz, el nuevo nodo es raíz. Si hay padre, se busca el padre y se adjunta el nuevo nodo en una de las cuatro posiciones específicas (hijo1, hijo2, hijo3, hijo4), verificando que no esté ocupada.

- **Eliminación de Nodos:**

Algoritmo (Poda): Si es la raíz, se vacía el árbol. Si no, se encuentra el padre, se desvincula el nodo a eliminar de ese padre (poniendo null en la referencia del hijo), y se eliminan recursivamente el nodo y todos sus descendientes de la lista global de nodos (nodos).

- **Recorrido en Anchura (BFS):**

Técnica: Cola (Queue).

Algoritmo: Encolar raíz. Desencolar nodo, procesar, encolar hijo1, hijo2, hijo3, hijo4 (si existen) en orden. Repetir hasta que la cola esté vacía.

- **Recorrido en Profundidad (DFS):**

Técnica: Pila (Stack) o recursión.

Algoritmo (Pila): Apilar raíz. Desapilar nodo, procesar, apilar hijo4, hijo3, hijo2, hijo1 (si existen) en ese orden. Repetir hasta que la pila esté vacía.

- **Recorrido Preorden (Raíz, Hijos):**

Técnica: Recursión.

Algoritmo: Procesar raíz. Llamada recursiva para hijo1, luego hijo2, luego hijo3, luego hijo4.

- **Recorrido "Inorden" (Hijo1, Raíz, Otros Hijos):**

Técnica: Recursión.

Algoritmo: Llamada recursiva para hijo1. Procesar raíz. Llamada recursiva para hijo2, luego hijo3, luego hijo4.

- **Recorrido Postorden (Hijos, Raíz):**

Técnica: Recursión.

Algoritmo: Llamada recursiva para hijo1, luego hijo2, luego hijo3, luego hijo4. Procesar raíz.

- **Matriz de Adyacencia:**

Técnica: Matriz 2D.

Algoritmo: Crear matriz $N \times N$ de ceros. Mapear etiquetas de nodo a índices. Para cada nodo, si tiene hijo1, hijo2, hijo3, o hijo4, poner un 1 en la celda [indice_padre][indice_hijo].

Desafíos encontrados y sus soluciones

Como problemas principales, el que más causo dolor de cabeza fue para dibujar el árbol ya que estaba como predeterminado en GridLayoutManager y tenía que cambiarle a border layout manager para dibujar el árbol como quería:

Property	Value
field name	panelArbol
Custom Create	<input type="checkbox"/>
Layout Manager	BorderLayout ▾
⊕ border	None
Horizontal Gap	0
Vertical Gap	0
⊕ Horizontal Size Policy	Can Shrink, Can Grow
⊕ Vertical Size Policy	Can Shrink, Can Grow
Horizontal Align	Fill
Vertical Align	Fill
Indent	0
⊕ Minimum Size	[-1, -1]
⊕ Preferred Size	[-1, -1]
⊕ Maximum Size	[-1, -1]
⊕ Client Properties	
background	<input type="checkbox"/> [43,45,48]
enabled	<input checked="" type="checkbox"/>
font	<default>
foreground	<input type="checkbox"/> [223,225,229]
toolTipText	

También fue un problema el tema de que hijos solo había izq y der, para ello tuve que pensar y cambiar para que puedan tener hasta cuatro hijos que pedía requerimiento.

Insertión de los nodos:

Arbol Customario Gráfico

Matriz Adyacencia

```

graph TD
    A((A)) --- H((H))
    A --- I((I))
    A --- J((J))
    H --- M((M))
    H --- N((N))
    H --- O((O))
  
```

Agregar Nodos:

Raíz: A

Hoja Izquierda: HJ04

Agregar Nodo

Dibujar Arbol

Recorridos

Recorrido en Anchura (BFS)

Recorrido en Profundidad (DFS)

Preorden

Inorden

Postorden

Eliminar Nodos: I

ELIMINAR

Nodos en el Arbol:

- A: A (485, 50)
- H: H (340, 110)
- I: I (437, 110)
- J: J (533, 110)
- K: K (630, 110)
- M: M (315, 170)
- N: N (364, 170)
- O: O (412, 170)

Activar Windows
Ve a Configuración para activar Windows.

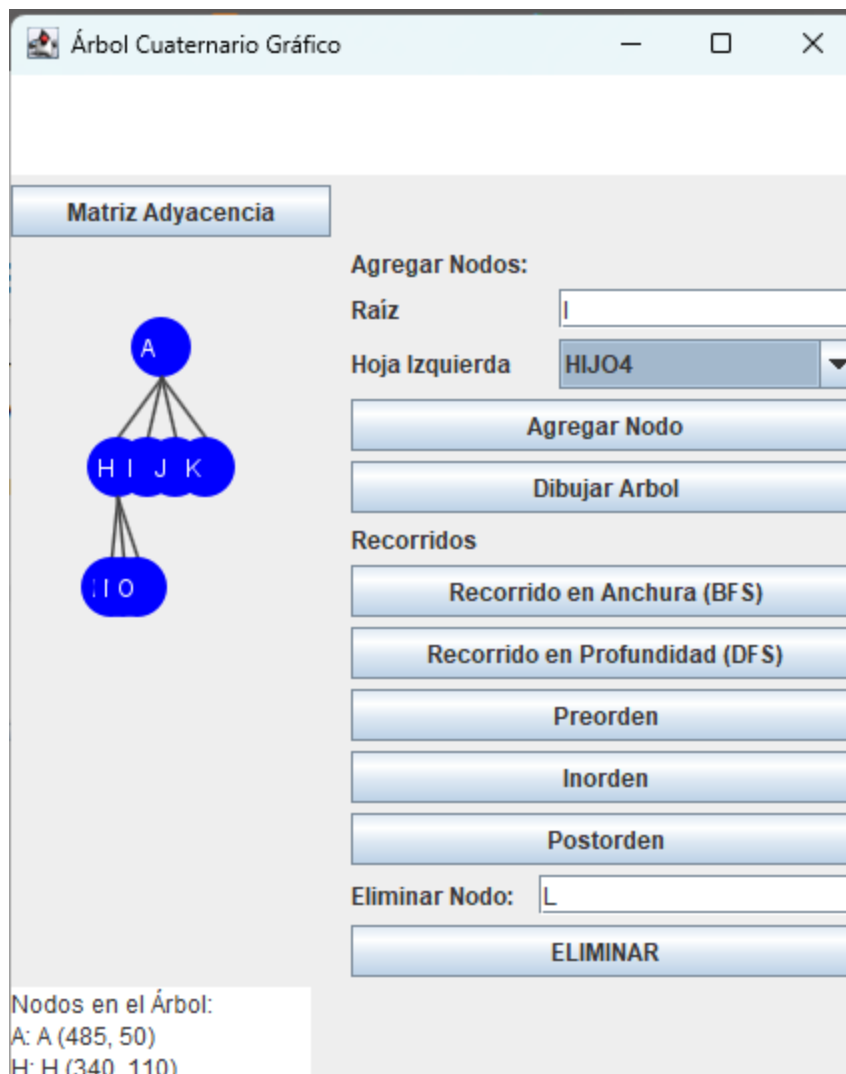
19°C
Mayorm, soleado

Buscar

ESP
LAA

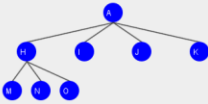
10:19
29/6/2023

1.- Primero escribo la raiz que deseo de la cual voy a crear los hijos con el combobox, e incerto los nodos. Ingreso la raiz I, de la cual eligo los hijo y voy agregando los nodos:



2.- De la misma manera para eliminar los nodos, escogo el nombre del cual quiero eliminar y elimino con el boton, añadi una validacion para estar seguro si eliminar el nodo elegido:

Matriz Adyacencia



```
graph TD; A((A)) --- H((H)); A --- I((I)); A --- J((J)); A --- K((K)); H --- M((M)); H --- N((N)); H --- O((O));
```

Agregar Nodos:

Raíz:

Hoja Izquierda:

Agregar Nudo

Dibujar Arbol

Recorridos

Recorrido en Anchura (BFS)

Recorrido en Profundidad (DFS)

Preorden

Inorden

Postorden

Eliminar Nudo:

ELIMINAR

Confirmar Eliminación

¿Está seguro de que desea eliminar el nodo 'O' y todo su subárbol?

Si No

Nodos en el Árbol:

A: A (485, 50)

H: H (340, 110)

I: I (437, 110)

J: J (533, 110)

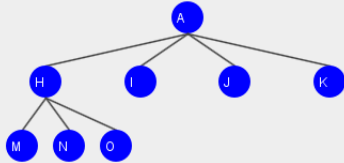
K: K (630, 110)

M: M (316, 170)

N: N (364, 170)

O: O (412, 170)

Matriz Adyacencia

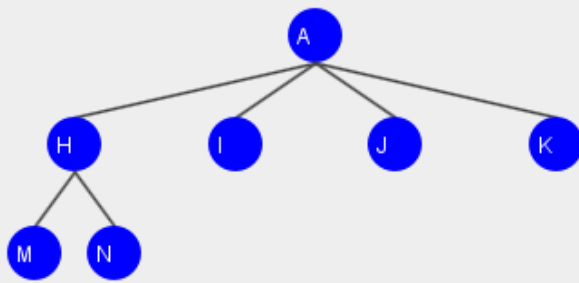


```
graph TD; A((A)) --- H((H)); A --- I((I)); A --- J((J)); A --- K((K)); H --- M((M)); H --- N((N)); H --- O((O));
```

Eliminación Exitosa

Nodo 'O' y su subárbol eliminados correctamente.

Aceptar



Recorridos del árbol trinario

Para los recorridos del arbol, los muestro en el textArea, al dar click en el recorrido deseado:

Nodos en el Árbol:
A: A (449, 50)
H: H (316, 110)
I: I (405, 110)
J: J (493, 110)
K: K (582, 110)
M: M (294, 170)
N: N (338, 170)
Preorden: A H M N I J K
Inorden (adaptado para cuaternario): H M N A I J K
Postorden: M N H I J K A