Búsqueda en profundidad y Búsqueda a lo ancho

Algoritmos útiles sobre grafos y árboles

1. Búsqueda en Profundidad (DFS - Depth-First Search)

La DFS explora tan lejos como sea posible a lo largo de cada rama antes de retroceder. En árboles binarios, se implementa naturalmente usando la **recursividad** (como en los recorridos preorden, inorden o postorden) o utilizando una **pila (Stack)** para la implementación iterativa.

A. Implementación Recursiva (Recorrido Preorden)

```
FUNCIÓN RECORRIDO_PREORDEN_DFS(Nodo)
    // Caso Base
    SI Nodo ES NULO ENTONCES
        RETORNAR
    FIN SI

    // 1. Visitar / Procesar la RAIZ
    PROCESAR(Nodo.Valor)

    // 2. Recursión a la IZQUIERDA
    RECORRIDO_PREORDEN_DFS(Nodo.Izquierdo)

    // 3. Recursión a la DERECHA
    RECORRIDO_PREORDEN_DFS(Nodo.Derecho)
FIN FUNCIÓN
```

Comentarios

La función recibe el puntero a nodo raíz.

PROCESAR es una forma generica de plantear que allí se trabaja con el dato. o se imprime o se realiza algo determinado.

IN ORDER y POST ORDEN se obtienen intercalando diferente las llamadas, o sea:

Para un recorrido Inorden, el "PROCESAR" iría entre el paso 2 y 3.

Para un recorrido Postorden, el "PROCESAR" iría después del paso 3.

B. Implementación Iterativa (Usando Pila)

```
FUNCIÓN RECORRIDO_DFS_ITERATIVO(Raiz)
    SI Raiz ES NULO ENTONCES
        RETORNAR
    FIN SI
    Pila = nueva Pila()
    APILAR(Pila, Raiz) // Empezar con la raíz
    MIENTRAS Pila NO ESTÁ VACÍA HACER
        // 1. Obtener y procesar el nodo actual (LIFO: Last-In, First-Out)
        NodoActual = DESAPILAR(Pila)
        PROCESAR(NodoActual.Valor)
        // 2. Añadir hijo derecho a la pila (para que se procese *después* del izquierdo)
        // Se añade primero el derecho, porque al ser LIFO, el izquierdo se desapilará primero.
        SI NodoActual.Derecho NO ES NULO ENTONCES
            APILAR(Pila, NodoActual.Derecho)
        FIN SI
        // 3. Añadir hijo izquierdo a la pila (para que se procese *primero*)
        SI NodoActual.Izquierdo NO ES NULO ENTONCES
            APILAR(Pila, NodoActual.Izquierdo)
        FIN SI
    FIN MIENTRAS
FIN FUNCIÓN
```

Comentarios

Como se ve, se utiliza un enfoque diferente para reemplazar a la recursión. Se utiliza una pila. En el ejemplo, la pila no esta implementada pero es exactamente como la hemos visto anteriormente.

2. Búsqueda en Anchura (BFS - Breadth-First Search)

La BFS explora todos los nodos en el **mismo nivel** antes de pasar al siguiente nivel. Esto asegura que se recorre el árbol **nivel por nivel**, de izquierda a derecha. Para lograr lo dicho se implementa usando una **Cola (Queue)**.

```
FUNCIÓN RECORRIDO_BFS(Raiz)
    SI Raiz ES NULO ENTONCES
        RETORNAR
    FIN SI
    Cola = nueva Cola()
    AGREGAR(Cola, Raiz) // Empezar con la raíz (FIFO: First-In, First-Out)
    MIENTRAS Cola NO ESTÁ VACÍA HACER
        // 1. Obtener y procesar el nodo actual
        NodoActual = FRENTE(Cola)
        ELIMINAR(Cola) // Quitar de la cola
        PROCESAR(NodoActual.Valor)
        // 2. Añadir el hijo IZQUIERDO a la cola
        SI NodoActual.Izquierdo NO ES NULO ENTONCES
            AGREGAR(Cola, NodoActual.Izquierdo)
        FIN SI

        // 3. Añadir el hijo DERECHO a la cola
        SI NodoActual.Derecho NO ES NULO ENTONCES
            AGREGAR(Cola, NodoActual.Derecho)
        FIN SI
    FIN MIENTRAS
FIN FUNCIÓN
```

Comentarios

Nuevamente, recordamos que esto es un pseudocódigo, en el proceso general, pueden utilizar una cola (hecha con una lista enlazada) o un simple array con el comportamiento de una cola.

Algoritmo para determinar el nivel de un nodo en un árbol

Algoritmos útiles sobre grafos y árboles

Ejemplo de aplicación: Determinar el nivel de un nodo.

El nivel o profundidad de un nodo se define como la cantidad de aristas (o bordes) desde la **raíz** hasta ese nodo. La raíz generalmente tiene un nivel de 0.

Este algoritmo utiliza una **búsqueda en anchura (BFS)** de forma implícita, manteniendo un contador para el nivel a medida que recorre el árbol desde la raíz.

Enfoque iterativo

```
FUNCIÓN ENCONTRAR_NIVEL(Raiz, ValorBuscado)
    // 1. Caso Base: El árbol está vacío
    SI Raiz ES NULO ENTONCES
        RETORNAR -1 // Indica que el nodo no fue encontrado
    FIN SI

    // 2. Inicialización
    NivelActual = 0 // La raíz está en el nivel 0
    Cola = nueva Lista() // Usaremos una cola para BFS
    AGREGAR(Cola, Raiz)

    // 3. Bucle principal de la Búsqueda en Anchura (BFS)
    MIENTRAS Cola NO ESTÁ VACÍA HACER
        // Obtener el número de nodos en el nivel actual
        TamañoNivel = TAMAÑO(Cola)

        // Procesar todos los nodos en este nivel
        PARA i DESDE 1 HASTA TamañoNivel HACER
            NodoActual = FRENTE(Cola) // Ver el nodo al frente
            ELIMINAR(Cola) // Quitar el nodo de la cola
```

```
// 4. Comprobación de éxito
            SI NodoActual.Valor ES IGUAL A ValorBuscado ENTONCES
                RETORNAR NivelActual // ¡Nodo encontrado!
            FIN SI

            // 5. Explorar hijos
            SI NodoActual.Izquierdo NO ES NULO ENTONCES
                AGREGAR(Cola, NodoActual.Izquierdo)
            FIN SI

            SI NodoActual.Derecho NO ES NULO ENTONCES
                AGREGAR(Cola, NodoActual.Derecho)
            FIN SI
        FIN PARA

        // 6. Pasar al siguiente nivel
        NivelActual = NivelActual + 1
    FIN MIENTRAS

    // 7. Nodo no encontrado
    RETORNAR -1 // El valor no se encontró en el árbol
FIN FUNCIÓN
```

Enfoque recursivo

```
// Función recursiva con el contador de nivel
FUNCIÓN ENCONTRAR_NIVEL_RECURSIVO(Nodo, ValorBuscado, NivelActual)
    // 1. Caso Base: Si el nodo es NULO, el valor no está en esta rama
    SI Nodo ES NULO ENTONCES
        RETORNAR -1
    FIN SI
    // 2. Caso Base: El nodo ha sido encontrado
    SI Nodo.Valor ES IGUAL A ValorBuscado ENTONCES
        RETORNAR NivelActual
    FIN SI
    // 3. Búsqueda en la rama izquierda
    NivelIzquierdo = ENCONTRAR_NIVEL_RECURSIVO(Nodo.Izquierdo, ValorBuscado, NivelActual + 1)

    // Si se encontró en la izquierda, se retorna inmediatamente
    SI NivelIzquierdo NO ES IGUAL A -1 ENTONCES
        RETORNAR NivelIzquierdo
    FIN SI
    // 4. Búsqueda en la rama derecha (solo si no se encontró en la izquierda)
    NivelDerecho = ENCONTRAR_NIVEL_RECURSIVO(Nodo.Derecho, ValorBuscado, NivelActual + 1)

    // 5. Retornar el resultado de la rama derecha (-1 si tampoco se encontró)
    RETORNAR NivelDerecho
FIN FUNCIÓN
```

Ejercicios

Algoritmos útiles sobre grafos y árboles

Escribir los algoritmos vistos en el lenguaje de programación C