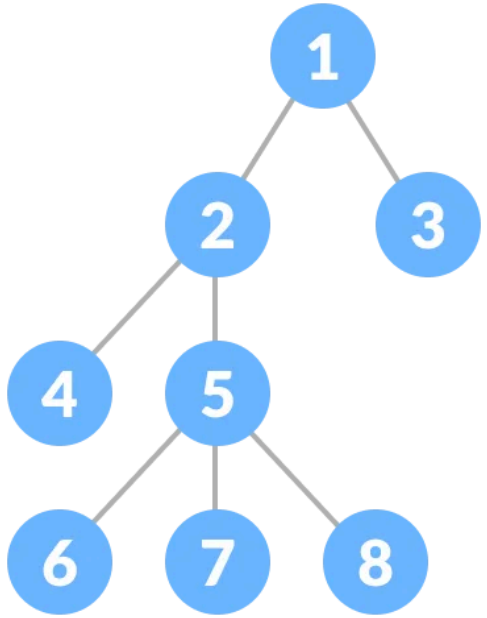
Ayudantía 6

Carlos Lagos - carlos.lagosc@usm.cl

¿Que es un Árbol?

Un árbol es una estructura jerárquica de datos que consta de un nodo raíz, nodos interiores con al menos un hijo, y nodos hojas sin hijos, utilizada para representar relaciones y organizar datos de manera jerárquica y estructurada.



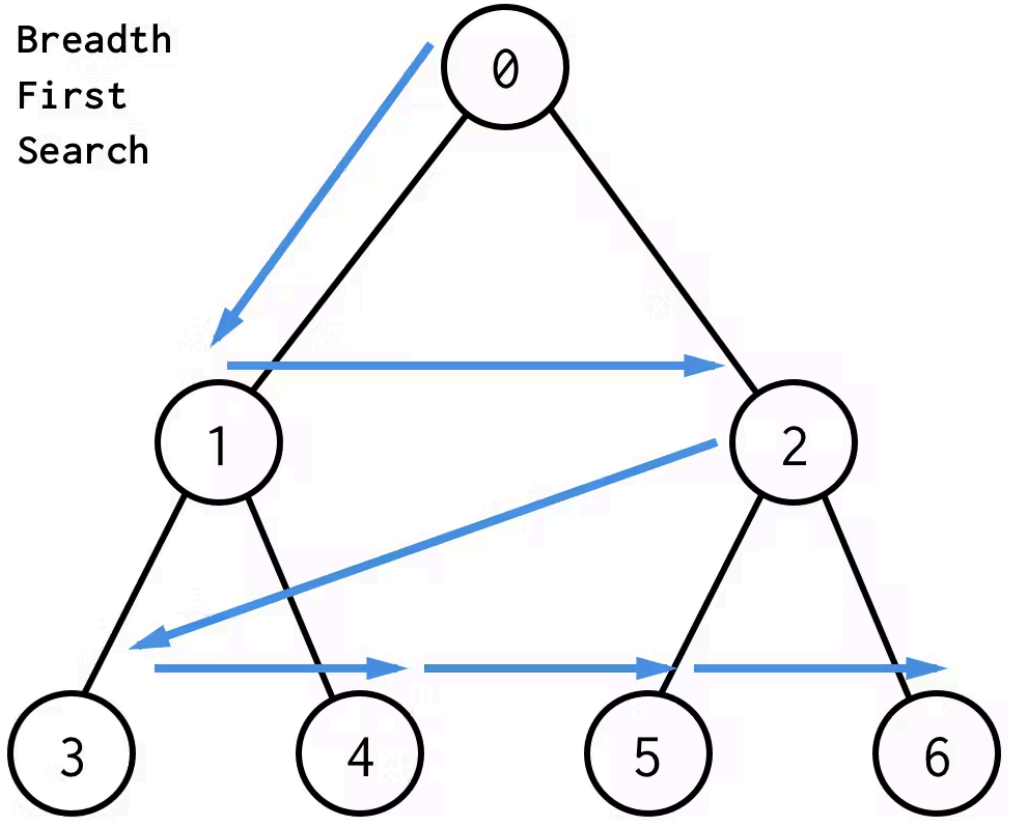
Recorrido en Anchura (BFS)

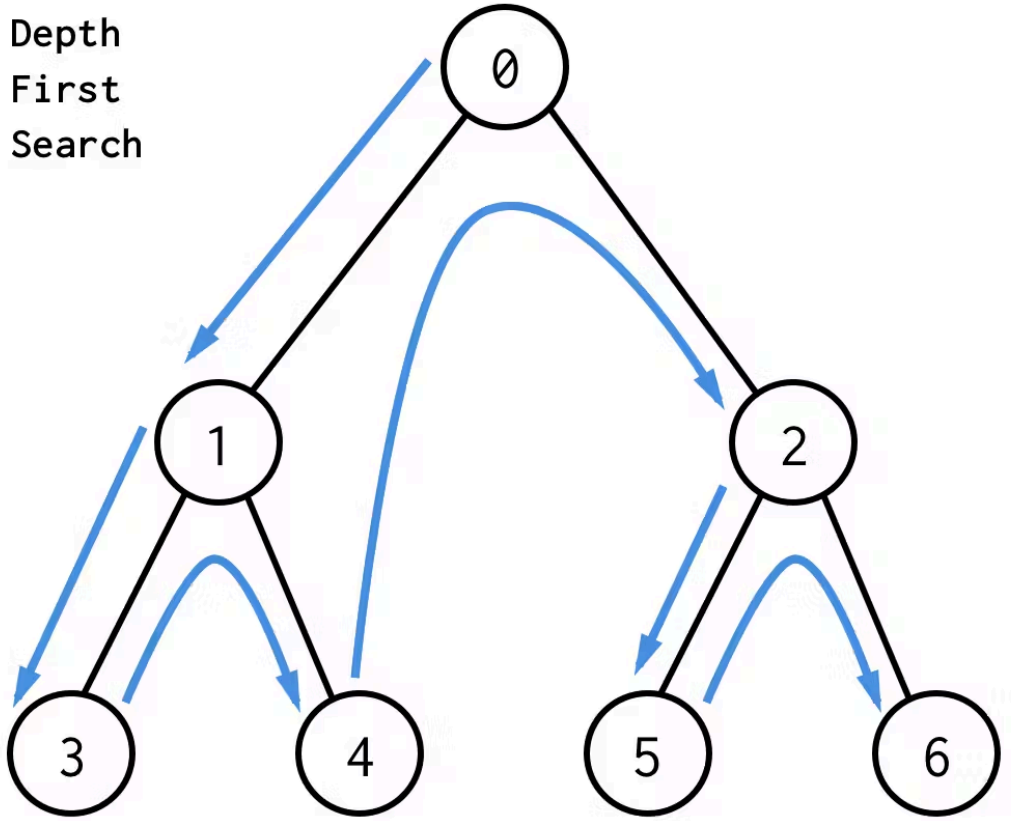
El recorrido en anchura de un árbol implica visitar primero la raíz y luego sus nodos hijos nivel por nivel, utilizando una estructura tipo cola para mantener el orden de visita y procesando cada nodo de manera secuencial.

Recorrido en Profundidad (DFS)

El recorrido en profundidad implica avanzar en el árbol hacia nodos más profundos en cada paso y, cuando se llega a una hoja o no hay más nodos disponibles en esa rama, se retrocede para continuar explorando otras ramas del árbol.

DFS vs BFS





Implementación de Árboles

1. **Lista enlazada para nodos hijos:**
   - Cada nodo guarda su primer hijo, y cada hijo guarda al siguiente hijo (su hermano), como si fuera una lista enlazada.
2. **Arreglo de punteros por nodo:**
   - Cada nodo tiene un arreglo que guarda los punteros hacia sus hijos.
3. **Uso de cualquier TDA lista:**
   - Similar al arreglo de punteros, pero utilizando cualquier TDA lista para manejar los punteros hacia los hijos.

Árboles binarios

El árbol binario es una variante de los árboles donde cada nodo puede tener hasta dos nodos hijos: uno izquierdo y otro derecho. Esta estructura sigue la definición general de un árbol, con la adición de la limitación de que pueden tener máximo dos hijos por nodo.

Recorridos en Árboles Binarios

Inorden

- Recorre primero el subárbol izquierdo, luego la raíz y finalmente el subárbol derecho.
- **Izquierda - Raíz - Derecha**

Recorridos en Árboles Binarios

Preorden

- Visita primero la raíz, luego el subárbol izquierdo y finalmente el subárbol derecho.
- **Raíz - Izquierda - Derecha**

Recorridos en Árboles Binarios

Postorden

- Recorre primero el subárbol izquierdo, luego el subárbol derecho y finalmente la raíz.
- **Izquierda - Derecha - Raíz**

Aplicaciones de los Recorridos

- **Inorden**: Imprime nodos en orden ascendente en un árbol de búsqueda binario.
- **Preorden**: Clonar árboles.
- **Postorden**: Libera la memoria de los nodos en un árbol.

Árboles Binarios de Búsqueda (ABB)

Los árboles binarios de búsqueda (ABB) son una variante de los árboles binarios que siguen una regla de orden en sus nodos. En un ABB, para cada nodo:

- Todos los nodos en el subárbol izquierdo tienen valores menores que el valor del nodo.
- Todos los nodos en el subárbol derecho tienen valores mayores que el valor del nodo.

Implementación de Árboles Binarios de Búsqueda (ABB)

Debido a la estructura de Árboles Binarios de Búsqueda (ABB), donde cada nodo puede tener como máximo dos hijos, se implementa de la siguiente manera:

- **Cada nodo guarda tres elementos:**
  - El valor del nodo.
  - Un puntero al hijo izquierdo.
  - Un puntero al hijo derecho.

Búsqueda en ABB

- Comienza en la raíz y compara el valor buscado con el valor del nodo actual.
- Si es igual, se ha encontrado el elemento.
- Si es menor, se busca en el subárbol izquierdo.
- Si es mayor, se busca en el subárbol derecho.
- Repite este proceso hasta encontrar el elemento o llegar a un nodo nulo.

Inserción en ABB

- Comienza en la raíz y compara el valor a insertar con el valor del nodo actual.
- Si es menor, se inserta en el subárbol izquierdo.
- Si es mayor, se inserta en el subárbol derecho.
- Repite este proceso hasta encontrar un lugar vacío (nodo nulo) donde insertar el nuevo elemento.

Eliminación en ABB

1. **Caso 1: Nodo sin hijos**
   - Se elimina el nodo directamente.
2. **Caso 2: Nodo con un hijo**
   - Se elimina el nodo y se reemplaza por su único hijo.
3. **Caso 3: Nodo con dos hijos**
   - Se busca el sucesor inmediato del nodo a eliminar (nodo con el menor valor en su subárbol derecho).
   - Se copia el valor del sucesor al nodo a eliminar.
   - Se elimina el sucesor, que ahora tiene un solo hijo o ninguno.

Ejercicios

- Grafica cómo sería la inserción de los siguientes elementos en el orden dado: $[7, 3, 1, 5, 11, 9, 13]$ y $[1, 3, 5, 7, 9, 11, 13]$ en un Árbol Binario de Búsqueda.
- Explica cuál sería la complejidad temporal de la inserción si se insertan los elementos del 1 hasta n en ese orden, es decir, $[1, 2, \ldots, n]$.
- Analiza la complejidad temporal al eliminar la raíz de un árbol binario que fue construido insertando los elementos en el siguiente orden: $[2, 1, n, 3, 4, \ldots, n-1]$.

Árbol AVL

- Un Árbol AVL es un tipo de árbol binario de búsqueda balanceado.
- La altura de los subárboles izquierdo y derecho difiere en máximo una unidad.
- Esto garantiza tiempos de búsqueda, inserción y eliminación eficientes.

Factores de Equilibrio

- Cada nodo de un AVL tiene un factor de equilibrio:
  - Factor de equilibrio = Altura del subárbol derecho - Altura del subárbol izquierdo.
- El factor de equilibrio debe estar en el rango [-1, 0, 1] para cada nodo.

Casos de Rotación

Rotación Simple Izquierda (LL)

- Desbalance: Factor de equilibrio = 2 en nodo X.
- Solución: Rotación simple a la izquierda en X para equilibrar el árbol.

Rotación Simple Derecha (RR)

- Desbalance: Factor de equilibrio = -2 en nodo X.
- Solución: Rotación simple a la derecha en X para equilibrar el árbol.

Casos de Rotación

Rotación Doble Izquierda-Derecha (LR)

- Desbalance: Factor de equilibrio = -2 en nodo X y 1 en nodo Y (hijo izquierdo de X).
- Solución: Rotación a la izquierda en Y seguida de rotación a la derecha en X.

Casos de Rotación

Rotación Doble Derecha-Izquierda (RL)

- Desbalance: Factor de equilibrio = 2 en nodo X y -1 en nodo Y (hijo derecho de X).
- Solución: Rotación a la derecha en Y seguida de rotación a la izquierda en X.

Ejercicios

- Dado el orden de inserción de los elementos $[10, 5, 3, 8, 15, 12]$, grafica cómo quedaría el árbol después de cada inserción.
- Explica cuál sería la complejidad temporal de la inserción si se insertan los elementos del 1 hasta n en ese orden, es decir, $[1, 2, \ldots, n]$.
- Analiza la complejidad temporal al eliminar la raíz de un árbol binario que fue construido insertando los elementos en el siguiente orden: $[2, 1, n, 3, 4, \ldots, n-1]$.

¿Qué es un Árbol 2-3?

- Es una estructura de datos en la que cada nodo puede tener 2 o 3 hijos.
- Permite almacenar más de un valor en cada nodo, lo que lo hace eficiente para ciertas operaciones.

Propiedades del Árbol 2-3

- Cada nodo puede tener 1 o 2 valores.
- Los nodos internos tienen 2 o 3 hijos.
- Todas las hojas están en el mismo nivel.

Gracias por su atención 😄