Unidad temática 4: Estructuras de datos no lineales

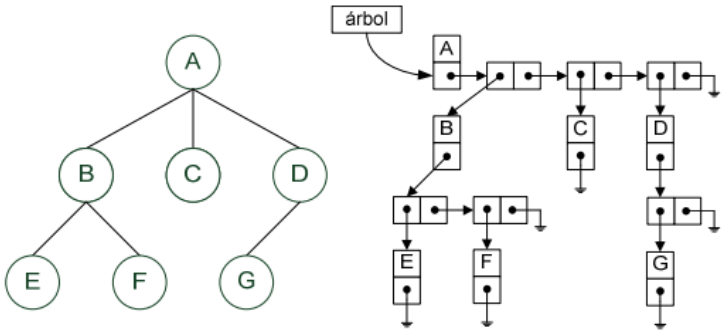
Nombre del alumno: Jiménez Téllez José Alfredo

Implementación de árboles de manera poco convencional

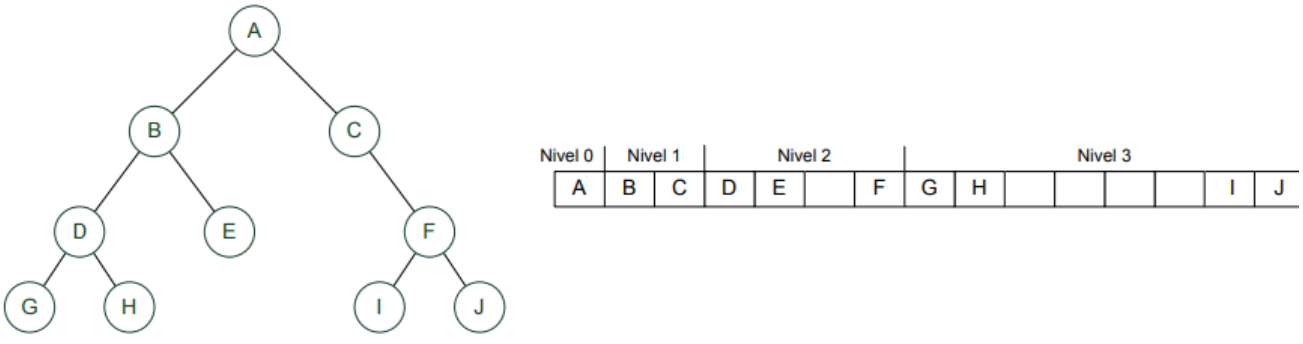
Los árboles son generalmente implementados a través de enlaces de nodos, compuestos de dos punteros que almacenarán a otros nodos o valores nulos, una filosofía muy parecida a la implementación de una estructura lineal dinámica. Sin embrago, para representar ciertos problemas, podemos abstraer los árboles de otra manera tal que nos permita representarlo de distinta forma a la convencional. Aquí algunos ejemplos:

-Implementación mediante una lista de listas: Es útil para problemas que requieran un árbol de soluciones, como en el caso de llevar el rastreo de la ruta que se toma en un método de búsqueda.

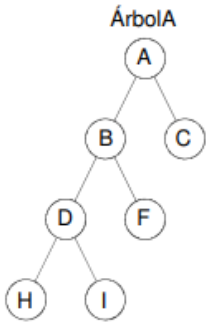
En esta implementación, cado nodo apunta a una lista enlazada, los nodos de dicha lista podrán tener a otro nodo enlazado que contiene a su vez, otra lista enlazada (cada nodo apunta a un nodo hijo).

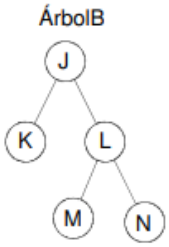


-Implementación como un array de elementos del tipo base: A través de un array, podemos almacenar los datos siguiendo un orden establecido. El primer elemento del array representará al nodo raíz, y los siguientes nodos se irán guardando en los lugares nodoPadre+1 y nodoPadre+2, a partir de ahí, nuestro array crecerá el doble cada que agreguemos un nuevo nivel de profundidad en el árbol.



-Implementación como un array de nodos: Empleando una matriz con 3 componentes para las columnas y n componentes para las filas, podemos almacenar valores que nos representen los enlaces existentes de un árbol. La columna 0 representará el índice del hijo izquierdo del nodo, la columna 2 será el índice del hijo derecho, y la columna 1(columna de en medio) nos representa la información del nodo en cuestión.





	hizq	raíz	hder
1	15	D	11
2			-1
3	7	A	9
4			-1
5	0	F	0
6	0	M	0
7	1	B	5
8	14	J	12
9	0	C	0
10	0	N	0
11	0	I	0
12	6	L	10
13			-1
14	0	K	0
15	0	H	0

ÁrbolA
3

ÁrbolB
8

Variación del Árbol B:

Árboles B+

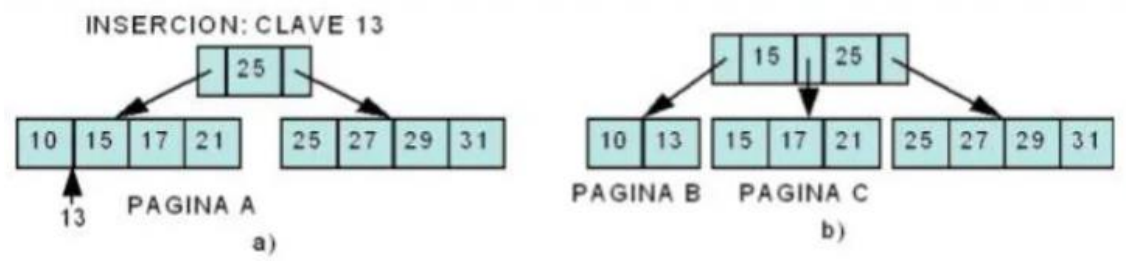
Nos proporcionan la misma ventaja de los Árboles B de contar con un acceso aleatorio y conservan la propiedad del recorrido secuencial rápido, pero todas las claves se encuentran en las hojas, duplicándose en la raíz.

Estos árboles requerirán el empleo de más memoria debido a la propiedad de tener duplicadas algunas claves de información, las claves de las páginas raíz e interiores se utilizarán únicamente como índices, pero para compensarlo, su recorrido secuencial será más rápido, ya que las hojas pueden vincularse, obteniendo una trayectoria secuencial para recorrer las claves del árbol.

Todas las claves se encuentran en hojas, duplicándose en la raíz y nodos interiores aquellas que resulten necesarias para definir los caminos de búsqueda.

En el método de inserción, el árbol B y B+ cambian, debido a que cuando se inserta una nueva clave en una página llena, ésta se divide en otras dos y lo que subirá a la página padre será una copia. Las llaves se almacenan en los nodos del último nivel del árbol y esos nodos se enlazan unos con otros formando una lista ligada que puede estar enlazada de forma sencilla o doblemente.

Al dividirse las páginas, la primera hoja contendrá (tomando a m como el orden) m/2 claves y la segunda (m/2) +1, y lo que subirá a la página antecesora será una copia de la clave central.



Para el borrado, habrá que considerar:

- Si al eliminar la clave (siempre en una hoja) el número de claves es mayor o igual a m/2 el proceso ha terminado. Las claves de las páginas raíz o internas no se

modifican, aunque sean una copia de la eliminada, pues siguen constituyendo un separador válido entre las claves de las páginas descendientes.

- Si al eliminar la clave el número de ellas en la página es menor que m/2 será necesaria una fusión y redistribución de estas tanto en las páginas hojas como en el índice.

Árboles de expansión de peso mínimo: Algoritmo de Kruskal

Desarrollado por el científico checo, Otakar Borukva en 1926. Nació a partir del intento de encontrar una red eléctrica que fuera eficiente para Moravia. Este algoritmo puede emplearse en el diseño de rutas aéreas para incrementar la eficiencia, e incluso se puede emplear para analizar correlaciones en sistemas financieros, entre otros más usos.

El algoritmo presentado a continuación encuentra la ruta en una red que emplee el menor peso para recorrer toda la red, o también puede encontrar la ruta con menos peso de un vértice dado a un punto específico. Por lo que podemos inferir que dicho árbol deberá contar con un peso establecido en las aristas/conexiones de vértice X a vértice Y.

Los pasos se pueden resumir de la siguiente forma:

1. Ordenar las aristas ascendentemente según su peso
2. La arista con menor peso será la arista inicial
3. De las aristas restantes, se selecciona la de menor peso, y si hay más de una de igual valor, se selecciona cualquiera de ellas.
4. Repetir paso 3 siempre y cuando la ruta no se repita
5. Finaliza una vez que tenemos en la ruta todos los vértices del grafo.

Árboles de decisión

Algunas de las aplicaciones de los árboles requieren de una diferente manera de encontrar datos, los árboles de decisión encuentran las posibles soluciones a un problema dado basadas en condiciones prestablecidas, y es muy empleado en machine learning para tareas de clasificación o regresión.

Se basan en el descubrimiento de patrones a partir de ejemplos. Su estructura consiste en un primer nodo llamado raíz y luego se descomponen el resto de los atributos de entrada en n ramas, planteando una condición que puede ser cierta o falsa. Suponiendo que las ramas n fueran igual a 2, **s**e bifurcaría cada nodo en 2 y volverían a subdividirse hasta llegar a las hojas que son los nodos finales y que equivalen a las respuestas: Si/No, Comprar/Vender, o lo que sea que estemos clasificando

Se puede decir entonces, que la tarea del aprendizaje inductivo consiste en inducir con los datos anteriores un mecanismo para inferir las clasificaciones de cada uno de los ejemplos a partir únicamente de las propiedades.

Gracias a la naturaleza inductiva de este método de búsqueda, podríamos predecir comportamientos futuros a partir de los comportamientos observados en el pasado. Por ejemplo, en el mundo de los créditos bancarios, a partir de los comportamientos de los clientes antiguos con respecto a la manera tan regular y temprana o no de sus pagos del crédito concedido, podemos inferir qué nuevos clientes tienen más probabilidad de hacer frente al pago de este y cuáles más probabilidad de dejarlo sin pagar.

Un nodo de decisión está asociado a uno de los atributos y tiene 2 o más ramas que salen de él, cada una de ellas representando los posibles valores que puede tomar el atributo asociado. De alguna forma, un nodo de decisión es como una pregunta que se le hace al ejemplo analizado, y dependiendo de la respuesta que dé, el flujo tomará una de las ramas salientes.

Un nodo-respuesta está asociado a la clasificación que se quiere proporcionar, y devuelve la decisión del árbol con respecto al ejemplo de entrada.

Ventaja de los árboles Bayer vs los AVL en términos de memoria

Es cierto que para un problema que requiera el manejo mayor volumen de datos, los árboles B proporcionan una herramienta más eficaz para el acceso de estos. Pero ¿Cómo interpretamos esto en términos de acceso a memoria?

En memoria interna el tiempo de acceso a n datos situados en distintas partes de la memoria es independiente de las direcciones que estos ocupen, mientras que en memoria externa es fundamental el acceder a datos situados en el mismo bloque para hacer que el tiempo de ejecución disminuya, si disminuimos el número de accesos a disco, lógicamente el tiempo resultante de ejecución de nuestra búsqueda se ve fuertemente recortado

Entonces, debido a que el tiempo de ejecución se incrementa junto con el número de acceso requeridos, un árbol binario requerirá un mayor número de accesos a disco para cargar un nodo en memoria y decidir entre dos posibles ramas. Mientras que un árbol con múltiples ramas y múltiples elementos en cada nivel nos proporciona la posibilidad de leer un número alto de datos sin necesidad de acceder a otro nodo