Árboles B+

Estructura de Datos Avanzada

Fernando Esponda

Lucía Lizardi c.u. 181036

Ana Cristina Sánchez Vázquez c.u. 179484

22 de abril del 2017

# Índice

[¿Qué es?](#_wnvduncma1ds) 3

[Características](#_bo82cvca1nvq) 4

[Operaciones](#_9eltgaopdh6m) 4

[Búsqueda](#_gkgforoirj7o) 4

[Inserción](#_z6byop5z6kw) 4

[Borrado](#_3woay9hik5eh) 4

[¿Para qué sirve?](#_6p0ue013yunq) 5

[Bibliografía](#_54n81ojovtn8) 6

# 

## 

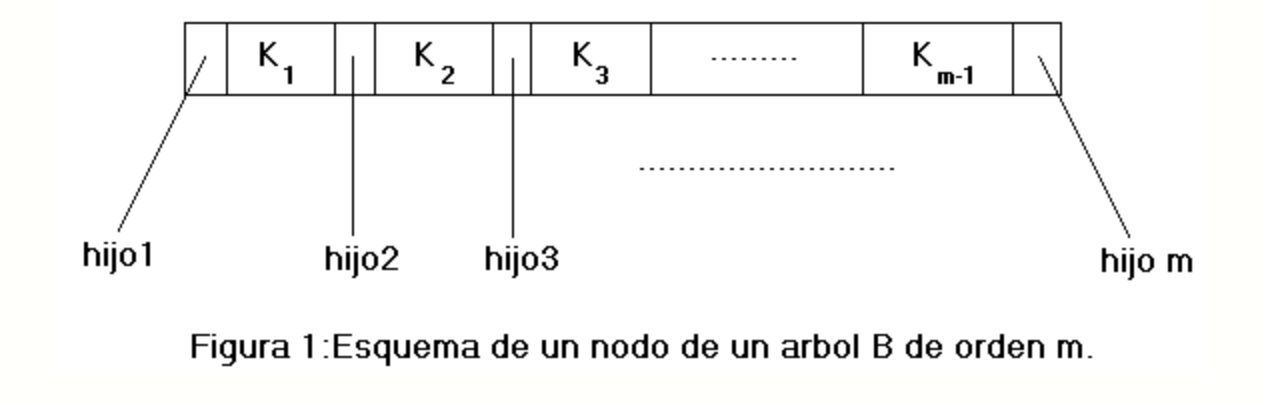
## ¿Qué es?

“Un árbol B+ es un tipo de **estructura de datos** que representa una colección de datos ordenados”

Estos árboles son una variación de los árboles tipo B, los cuales a diferencia de un árbol binario (que está formado por nodos y pueden tener hasta dos hijos), pueden tener x número de hijos. Dichos árboles también tienen una “clave”, que diferencía al nodo, llamémosla k. Siempre teniendo en cuenta que k1<k2<k3<...<km-1(por ser una colección de datos ordenados). También, los hijos que cuelgan del hijo 1 tienen claves menor a k1, en el segundo hijo, mayores a k1 y menores a k2, de esta forma hasta Km-1.

Un árbol de tipo B es más eficiente que uno binario de búsqueda para grandes cantidades de datos principalmente porque los árboles de múltiples ramas tienen una altura menor que un árbol de búsqueda binaria.

Ejemplo de un árbol B:



Como se ha visto en clase, algo que se debe buscar a la hora de programar es minimizar el tiempo de ejecución de un código al menor posible. En el curso hemos encontrado soluciones a esto por medio de diferentes estructuras de datos. Con esto en mente, el problema que se busca resolver con un árbol B está enfocado hacia las bases de datos y sistemas de archivos ya que, como fue mencionado, son más eficientes para grandes cantidades de datos.

Dicho lo anterior. Un árbol **B+** es una variación de un árbol B los cuales conservan la propiedad de acceso aleatorio rápido. Permiten un recorrido secuencial rápido. Todas las claves se encuentran en hojas, duplicándose en la raíz y en los nodos interiores necesarios para definir los caminos de búsqueda, las hojas son las que contienen a los elementos y los nodos-hoja forman una lista doblemente enlazada. Las hojas se pueden vincular para facilitar el recorrido secuencial rápido.**Se busca reducir el número de accesos buscando que dichos accesos tengan elementos de mayor tamaño.**

Su principal característica es que todas las claves se encuentran en las hojas. Es por esto que ocupan más espacio que los árboles B, ya que las claves pueden repetirse. Las claves de la raíz y de los nodos interiores se utilizan únicamente como índices.

## Características

* La raíz almacena al menos un dato y máximo m-1 datos.
* La raíz tiene al menos dos hijos.
* Los nodos interiores tienen mínimo (m-1)/2 datos y máximo m-1 datos.
* Las hojas tienen la misma altura.
* La información contenida en el árbol está ordenada.
* La información se encuentra en las hojas, por lo que los datos de nodos intermedios se encuentran duplicados.
* Tiene una altura que se define en el peor de los casos como logm(n). Siendo m el número máximo de hijos que un nodo puede tener.

## Operaciones

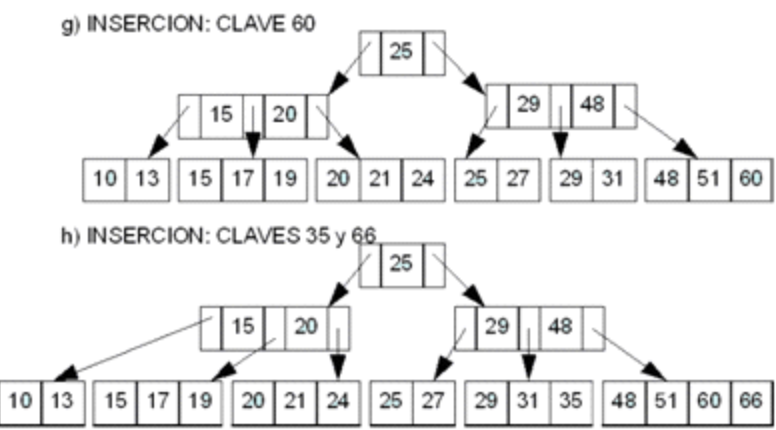
Al igual que en otras estructuras previamente estudiadas, los árboles B+ cuentan con 3 operaciones básicas: búsqueda, inserción y borrado de algún elemento.

### Búsqueda

La búsqueda, a diferencia de otros tipos de estructuras, no se detiene cuando se encuentra el elemento en algún nodo, sino que sigue hasta la hoja donde se encuentra el elemento. Esto se hace siguiendo por la rama derecha del nodo en donde se encuentra el elemento.

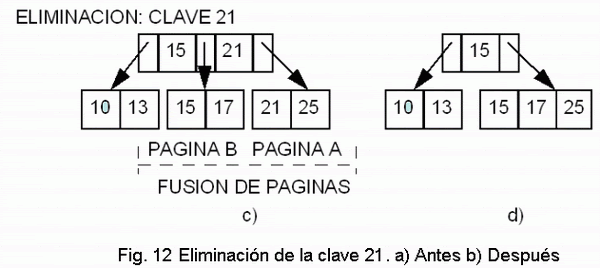
### Inserción

Cuando un nuevo elemento se inserta en un nodo lleno, el nodo se divide en dos (como un árbol 2-3). Uno de los nodos contendrá m/2 elementos y el otro m/2+1. El elemento central del nodo original sube al nodo papá.



### Borrado

Se deben considerar varias cosas:

* Que al borrar un elemento, el número de hojas que queden sea m/2<=x<=m-1
* Que los elementos de los nodos internos no se modifiquen.
* Cuando el número de hijos de un nodo cambia a un número no válido, se debe hacer una redistribución (tal como se vio en los árboles 2-3)

## 

## 

## 

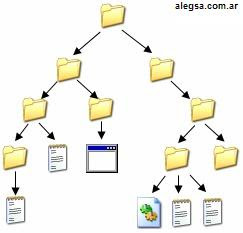
## 

## ¿Para qué sirve?

Como fue mencionado anteriormente, estos árboles fueron diseñados principalmente para los sistemas de almacenamiento masivo ya que el tiempo de acceso a datos en ellos, es mayor que el tiempo de transferencia. Es decir, la búsqueda es más costosa que la lectura.

EJEMPLOS:

* En una base de datos:
  + Supongamos que una empresa necesita almacenar 2 millones de datos.
  + Si usaramos un árbol AVL su altura sería de log2(200000)21
  + Si se usa un árbol B+, de orden aproximadamente 1000, tendría una altura de 2-3
  + El acceso a disco, (que es lo que se busca minimizar) sería de 1 o 2 accesos, por la forma en la que se diseña. (Ejemplos gráficos más abajo).
* Sistema de archivos:
  + En general, para un sistema de archivos, el uso de un árbol B+, es adecuado.
  + Porque se almacenan datos únicamente en las hojas.



* NTFS ( New Technology File System):
  + Este por ser un sistema de archivos.
  + Un ejemplo es el caso del uso de NTFS para guardar directorios.
  + Por consecuencia, estos se usan al momento de buscar archivos.

## Bibliografía

* ÁRBOLES B . (n.d.). Retrieved April 19, 2020, from http://decsai.ugr.es/~jfv/ed1/tedi/cdrom/docs/arb\_B3.htm
* Arboles B - tutoriasarboles. (n.d.). Retrieved April 20, 2020, from https://sites.google.com/site/tutoriasarboles/arboles-b-y-b
* Seguir. (n.d.). Arboles B y Arboles B . Retrieved April 19, 2020, from https://es.slideshare.net/neltherdaza/arboles-b-y-arboles-b
* Foros aprenderaprogramar.com Didctica y divulgacin de la programacin. (n.d.). Retrieved April 20, 2020, from https://aprenderaprogramar.com/foros/index.php?topic=1436.0
* Estructuras de Datos y Algoritmos. (n.d.). Retrieved April 21, 2020, from https://www.infor.uva.es/~cvaca/asigs/doceda/tema4.pdf
* Arboles B ´. (n.d.). Retrieved April 20, 2020, from http://dis.um.es/~ginesgm/files/doc/aed/sec4.4.pdf