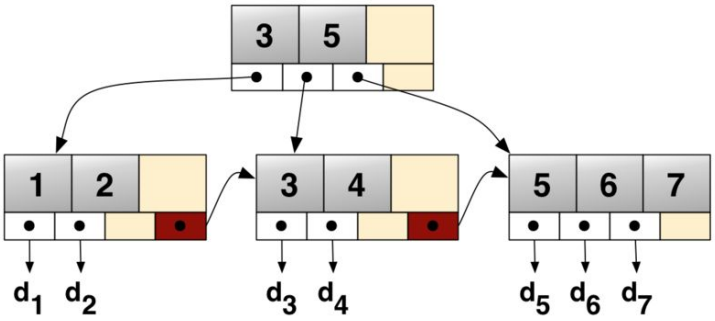
Rodrigo Barrera CU: 181935
Axel Flores CU: 181218

Árboles B+

¿Qué son?

Los árboles B+ son un tipo de estructuras de datos, representa una colección de datos ordenados de manera que se permite la inserción y eliminación eficiente de elementos.



Para entender más a fondo el comportamiento de un árbol B+, es importante tener claro que es un árbol B ya que los árboles B+ son variaciones de estos. Un árbol B son árboles de búsqueda los cuales tienen nodos capaces de tener varios hijos como se puede ver en la imagen adjunta. Entonces, los árboles B+ conservan la propiedad de acceso aleatorio rápido y permiten un recorrido secuencial rápido. Los árboles B+ ocupan más espacio que los B y la mayor diferencia es que en estos árboles, toda la información se encuentra almacenada en las hojas. En la raíz y en las "páginas internas" se encuentran almacenados índices o claves para así, llegar a un dato.

Características principales

- El árbol, es de un factor de ramificación u orden (b) de acuerdo con el cual el que se acomodan los datos.
- La raíz puede guardar mínimo un dato y máximo b-1 datos.
- La raíz debe tener mínimo dos hijos.
- Los nodos internos tienen entre (b-1)/2 y b-1 datos.
- Todas las hojas están a la misma altura.
- Los datos están ordenados.
- Todos los datos están en las hojas, pero en los nodos internos existen sus copias.

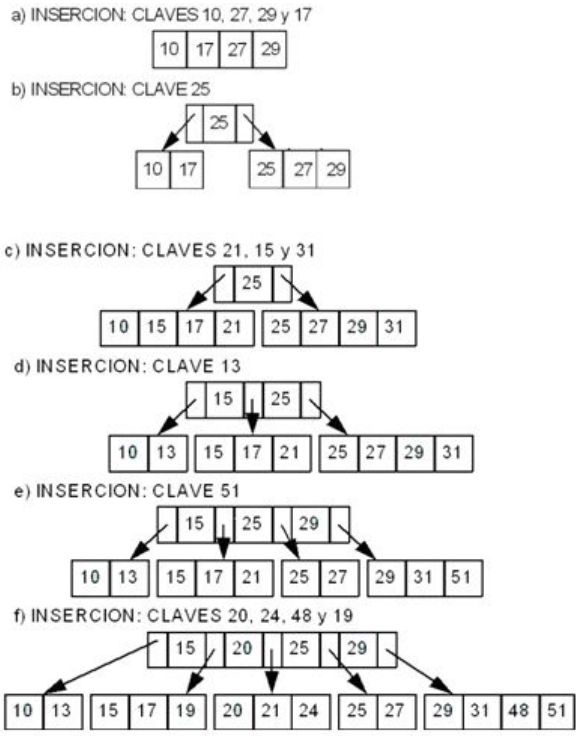
Operaciones

Al igual que en los árboles B, los árboles B+ tienen las mismas operaciones, estas siendo:

- **Búsqueda**
  - La búsqueda es similar a la búsqueda en los árboles B.
  - Se recorre el árbol de misma manera que en un árbol B, cuando se encuentra el la clave se asegura que esta se encuentre en un nodo que no sea ni raíz ni interior,

debe continuar la búsqueda con el nodo apuntado por la rama derecha de dicha clave.

- **Inserción**
  - Se ubica la raíz.
  - Se evalúa si es una hoja.
    - Si si es verdadero, se checa que no sobrepase el límite de datos.
    - Si es falso, se divide ese nodo a la mitad y se sube una copia a la mediana del nodo padre, si el padre se encuentra lleno, se aplica el mismo método hasta llegar a que la raíz, la altura del árbol aumenta en 1.
  - Si no es hoja, se compara el elemento a insertar con cada uno de los valores almacenados para encontrar la página descendiente donde se pueda continuar con la búsqueda.
  - Se regresa a la búsqueda.



- **Eliminación**
  - Se ubica el valor que se borrar. Si no está, ya terminé.
  - Si encontré el elemento, únicamente se borra de las hojas, pero no de la raíz y los nodos internos, no importan que tengan una copia.
  - Una vez que se borró de la hoja, se tienen dos casos.
    - Si el número de elementos de la hoja es mayor o igual a b/2, entonces ya acabé y no se modifica ningún nodo.
    - Si el número de elementos de la hoja es menor a b/2, entonces de debe de hacer una redistribución de los nodos, tanto hojas como internos.

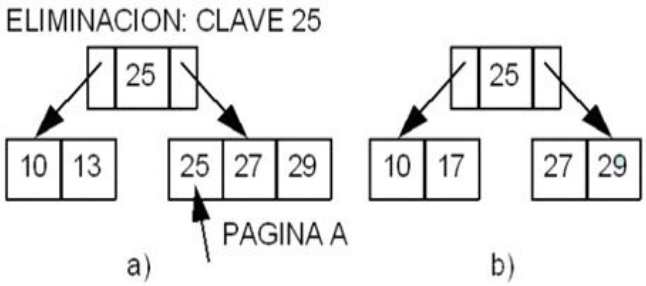


Fig. 10 Eliminación de la llave 25. a) Antes b) Después

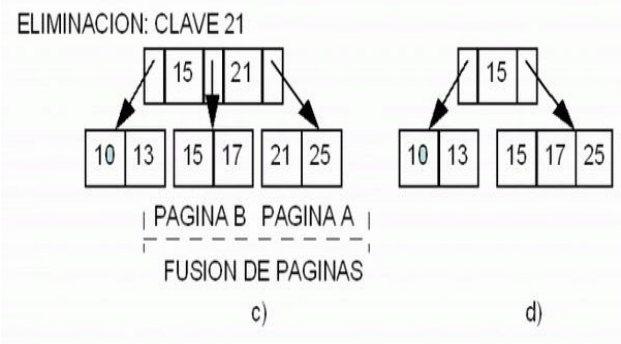


Fig. 12 Eliminación de la clave 21. a) Antes b) Después

Diferencias con las Estructuras vistas antes en clase:

Árboles Binarios: a diferencia de estos árboles, en un árbol B+cada nodo del árbol puede tener más de dos nodos hijos y más de un elemento guardado por nodo.

Árboles Binarios de Búsqueda: misma diferencia que los árboles binarios sencillos, pero ambos comparten el orden de los elementos i.e los menores a la izquierda y los mayores a la derecha.

Árboles AVL: mismas diferencias y semejanzas que con los árboles binarios de búsqueda, pero ambos se mantienen "balanceados" por diferentes métodos durante la inserción y eliminación.

Skip List: a diferencia de los árboles B+, una skip list es una estructura lineal y probabilística en lugar de determinista.

Heaps: en clase se implementaron los heaps con heaps con un arreglo, además los árboles B+ no se van llenando en el mismo orden no importa el elemento.

Aplicaciones

Los DBMS utilizan los árboles B+, para indexar los valores dentro de la base de datos y así poder acceder a la información de manera más rápida. Esto se debe a que las hojas se guardan en memoria secundaria mientras que los nodos internos, que contienen los referencias a los datos, se encuentran en la memoria primaria.

Los árboles B+ se utilizan para hacer más eficiente el funcionamiento de los sistemas de archivo en las memorias flash. Este tipo de memorias son utilizadas por los dispositivos móviles como método de almacenamiento. Este tipo de memorias se utiliza también en USBs y unidades de estado sólido.

Además, otra aplicación para la que los árboles B+ sirven es en el hecho de que ayudan a tener operaciones más rápidas en el disco. Este proceso es similar a una skip list; basándonos en la búsqueda ya descrita previamente. Al no tener que recorrer todos los elementos para realizar la búsqueda, sólo se revisa si es mayor o menor; esto genera una eficiencia ya que esto presenta una situación donde se reduce el tiempo de búsqueda en el disco; así se evita tener que leer el disco completo cada vez que se busca un elemento.

Referencias:

- https://estructurasite.wordpress.com/arbol-b-3/
- http://decsai.ugr.es/~jfv/ed1/tedi/cdrom/docs/arb_B3.htm
- https://garciagregorio.webcindario.com/ed/arboles_clasificacion.pdf
- https://www.youtube.com/watch?v=aZjYr87r1b8
- https://www.slideshare.net/neltherdaza/arboles-b-y-arboles-b
- https://www.javatpoint.com/b-plus-tree
- https://www.softwaretestinghelp.com/b-tree-data-structure-cpp/
- https://sites.google.com/site/tutoriasarboles/arboles-b-y-b
- https://www.researchgate.net/publication/225196332_An_Improved_B_Tree_for_Flash_File_Systems
- https://www.tutorialcup.com/dbms/b-tree.htm
-