Investigación Árboles B+

Andres Quevedo
181336

Estructuras de Datos Avanzadas – Abril 22, 2020

¿Qué Son?

Los árboles B+ son otro tipo de árbol de búsqueda similares a los árboles B o árboles 2-3 que hemos estudiado previamente ya que consisten de una estructura de nodos interconectados, donde cada nodo tiene una cantidad finita de hijos y un padre. Estos árboles además se caracterizan porque el hijo izquierdo de un nodo N tiene apuntador comparativamente menor a la de N y el hijo derecho de N tiene un apuntador mayor. Esto facilita el procedimiento de búsqueda porque la estructura tiene un cierto orden.

Diferencias con otras estructuras

Los árboles B tienen toda la estructura mencionada previamente y además tienen la característica que cada nodo contiene la información relacionada a la referencia de dicho nodo, es decir, a través de todo el árbol la información está contenida en cada nodo. La diferencia con los árboles B+ es que esto no sucede ya que solo los *nodos hoja* contienen la información del árbol, todos los demás nodos llamados *nodos internos* sólamente contienen el apuntador a sus nodos parientes. Adicionalmente el árbol B+ se relaciona con el árbol 2-3 que igual estudiamos porque cada nodo de ambos árboles en realidad es un conjunto de varios nodos con su propia cantidad de apuntadores.

Esta nueva estructura que implementan los árboles B+ funciona para bajar el tiempo de búsqueda dentro de un árbol ya que disminuye el tamaño de todos los nodos que el algorítmo tiene que recorrer hasta encontrar su destino, y solo se preocupa de leer la información cuando llega a un *nodo hoja*.

Implementación

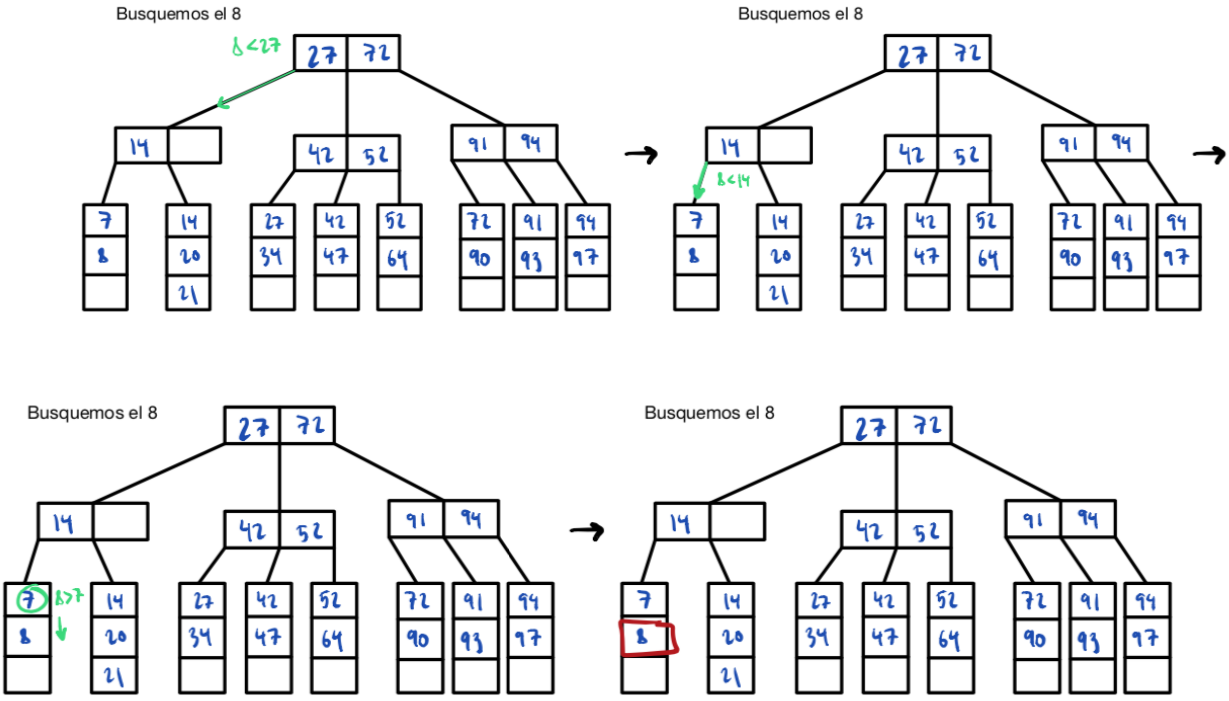
Como mencionamos previamente el árbol B+ contiene toda su información dentro de los *nodo hoja* y los demás nodos denominados *nodos internos* sólo contienen apuntadores a otros nodos. Para mantener la integridad del árbol también se piden ciertas restricciones:

1. Los *nodos internos* tienen máximo M hijos y mínimo $\lceil M \rceil$.
2. Todos los nodos tienen máximo L elementos y mínimo $\lceil L \rceil$.
3. Los nodos dentro de cada paso del árbol está ordenado de menor a mayor.
4. Los elementos de los *nodos internos* son el valor más pequeño de cada uno de sus hijos empezando por el segundo hijo.

Búsqueda

La búsqueda dentro de un árbol B+ es sencilla porque se parece mucho a las búsquedas realizadas en otros árboles. Consiste en empezar en la raíz del árbol y comparar el apuntador del nodo que estamos buscando con los nodos dentro de la raíz. Cuando encontramos el apuntador que corresponde, es decir el apuntador donde el nodo buscado sea más grande que el nodo a la izquierda pero más chico que el nodo a la derecha, entonces viajamos a través de ese apuntador hacia el siguiente nodo. Repetimos este procedimiento hasta llegar a un nodo que no tiene hijos lo cual es un *nodo hoja*.

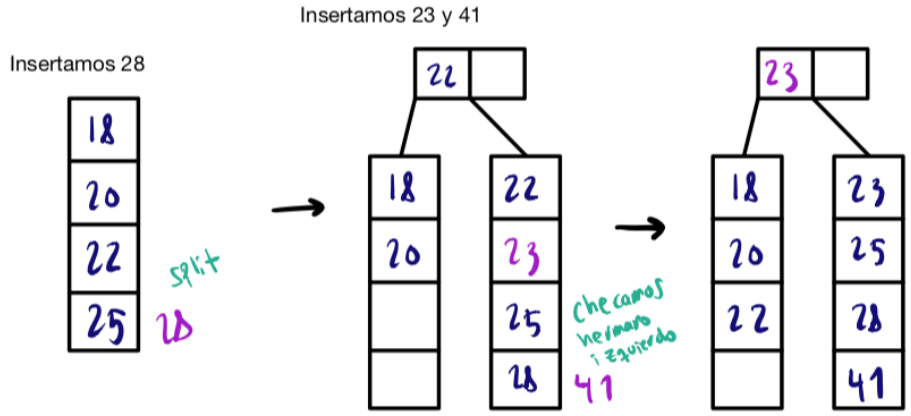
Dentro de este *nodo hoja* esta contenido el nodo que buscamos, o bien esta el lugar donde estaría el nodo, en el caso que este no exista. Adicionalmente como el *nodo hoja* esta ordenado entonces encontrar el nodo que buscamos o su ausencia será de manera trivial.



Inserción

Las inserciones en un árbol B+ son un poco más complejas que la búsqueda y consisten en encontrar el lugar donde el nuevo nodo debería ir (de la misma manera que buscamos un nodo) y de ahí el algoritmo decide que procede de acuerdo a los siguientes pasos. Cabe notar que cada inserción se hará forzosamente dentro de un *nodo hoja*.

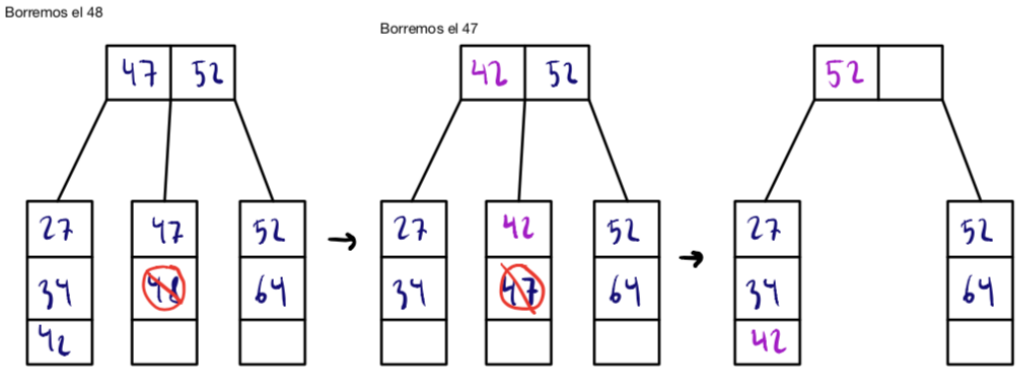
- Si el *nodo hoja* correspondiente tiene espacio entonces el nodo se inserta en el lugar cuidando el orden del *nodo hoja*
- Si el *nodo hoja* correspondiente está lleno, entonces:
    1. Se checa el hermano izquierdo para ver si hay espacio y si sí entonces el elemento más chico del *nodo hoja* actual se pasa al hermano izquierdo.
    2. Si el hermano izquierdo está lleno entonces checamos el hermano derecho y si tiene espacio entonces pasamos el elemento maás grande del *nodo hoja* actual al hermano derecho.
    3. Si ambos hermanos están llenos entonces necesitamos dividir el *nodo hoja*. Esto se hace buscando el elemento medio del *nodo hoja* actual, creando un nuevo *nodo hoja*, y finalmente insertamos dentro del nuevo *nodo hoja* el elemento medio y todos los elementos mayores a este de tal manera que quedamos con dos *nodos hoja* medio llenos.



Eliminación

La eliminación de un nodo dentro de un árbol B+ comienza buscando el nodo que deseamos borrar. Una vez más este nodo forzosamente estara dentro de un *nodo hoja* y la eliminación sigue los siguientes pasos:

- Si el elemento que se va a borrar no existe dentro del árbol entonces no se hace nada.
- Si si existe entonces se borra el nodo y se checa la cantidad de elementos dentro del *nodo hoja*. Si la cantidad es mayor al mínimo entonces no se hace nada, si sí entonces:
  1. Se checa el *nodo hoja* izquierdo para ver si se puede sacar un elemento de ahí. Si sí se puede entonces se saca el elemento más grande del hermano izquierdo y se pone en el *nodo hoja* actual.
  2. Si el hermano izquierdo no tiene suficientes elementos para dar entonces se juntan los dos *nodos hoja* en uno solo.
  3. Si no existe el hermano izquierdo entonces se checa el hermano derecho para ver si puede dar un elemento. Si sí se puede entonces se saca el elemento más pequeño y se coloca dentro del *nodo hoja* actual.
  4. De igual manera si el hermano derecho tiene muy pocos elementos entonces se hace un merge.



Ejemplos

Bases de Datos

Una aplicación para los árboles B+ es el uso de estas aplicaciones dentro de bases de datos, como por ejemplo las bases de datos controladas por SQL o MySQL. La ventaja de utilizar específicamente estas estructuras es que mantiene los datos ordenados de manera secuencial lo cual facilita y apresura viajar a través del árbol, minimiza la cantidad de lecturas a la memoria, mantiene un balance de distancia entre la raíz y cualquier nodo sin importar la cantidad de datos y finalmente un árbol B+ puede lidiar con una cantidad enorme de datos lo cual es muy útil cuando se trata de una base de datos. Los árboles B+ también son usados cuando se necesita leer datos del disco duro que tarda mucho, y esta estructura permite que se acelere ese proceso.

Búsqueda de Archivos

Un ejemplo concreto de como se usan los árboles B+ en la vida real es un problema expuesto en la página *codeproject.com* donde el problema era que se necesitaba buscar un método para acceder archivos contenidos dentro de folders que podían guardar un máximo de N archivos. Claramente un árbol B+ fue una solución perfecta a este problema ya que cada nodo ahora podía representar un archivo con una capacidad limitada de N elementos y solo se necesitaba crear apuntadores entre estos folders. Los algoritmos del árbol B+ hicieron el resto organizando los archivos mientras eran insertados para ser buscados de una manera muy eficiente posteriormente.

Search Engines

La última aplicación de estas estructuras de datos son su utilidad en los motores de búsqueda o *search engines* al momento de lanzar consultas en búsqueda de resultados relacionados a lo que el usuario quiere buscar. Encontre un ejemplo de un artículo de la Universidad de Mumbai donde se necesitaba buscar una manera de recorrer una base de datos para encontrar la información demandada por un usuario utilizando un *search engine*, en búsqueda de las palabras claves de la búsqueda. En este caso se utilizo un árbol B+ para llevar a cabo este proyecto y efectivamente encontrar la información requerida y recorrer la base de datos con eficiencia.

Referencias

[1] MOGA, Mihai. “Introduction to B-Trees: Concepts and Applications.” CodeProject, CodeProject, 15 Aug. 2014, www.codeproject.com/Articles/808055/Introduction-to-B-trees-Concepts-and-Applications.

[2] Bijral, Simran and Mukhopadhyay, Debajyoti. (2014). Efficient Fuzzy Search Engine with B -Tree Search Mechanism. 10.1109/ICIT.2014.19.

[3] Saagnik Adhikary, et al. “Introduction of B+ Tree.” GeeksforGeeks, 9 Apr. 2020, www.geeksforgeeks.org/introduction-of-b-tree/.