**Introduction of B+ Tree**

Last Updated : 08 Mar, 2024

**Nodes in B+ Tree**

In B+ tree, all the data is pointed to by the leaf nodes. Internal nodes are only used for the pointing purposes. All the leaf nodes are in the same level.

* **Internal Nodes**: These contain only keys to guide the search but not actual data values. These keys are used to search for the elements inside the B+ trees. They have pointers to child nodes.
* **Leaf Nodes**: These contain keys and actual data values. They are linked together to form a linked list, which allows for efficient range queries and ordered traversal.

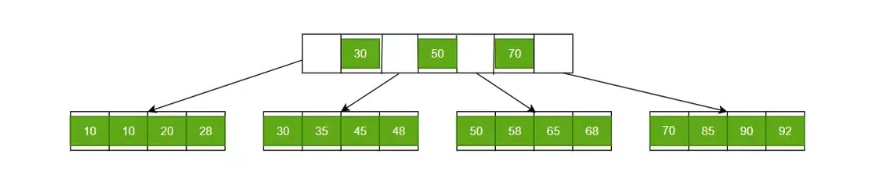
The number of children a node can have is dependent on the order of the B+ tree.

**Order of B+ Tree**

The order ***m*** of a B+ tree defines the maximum number of children that an internal node can have. An internal node can have at most ***m*** children and at least***m/2***childrens except the root, which can have fewer.

The root node of the B+ tree must have a minimum of two children and must contain at least one key.

**Structure of B+ Trees**



B+ Trees contain two types of nodes:

* **Internal Nodes:**Internal Nodes are the nodes that are present in at least n/2 record pointers, but not in the root node,
* **Leaf Nodes:**Leaf Nodes are the nodes that have n pointers.

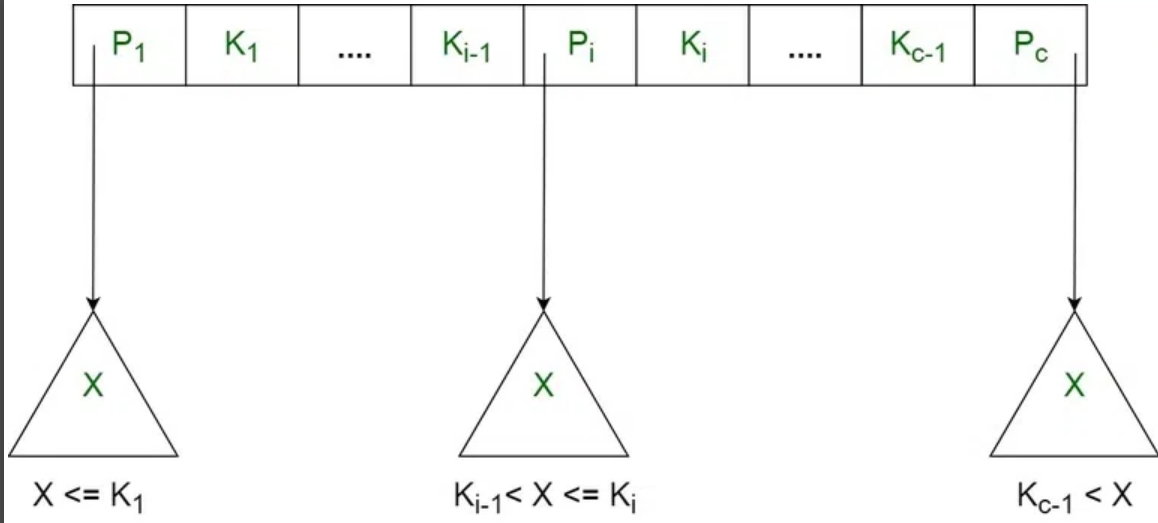
**The Structure of the Internal Nodes of a B+ Tree of Order ‘a’ is as Follows**

* Each internal node is of the form: <P1, K1, P2, K2, ….., Pc-1, Kc-1, Pc> where c <= a and each **Pi is a tree pointer (i.e points to another node of the tree)** and, each **Ki is a key-value** (see diagram-I for reference).
* Every internal node has : K1 < K2 < …. < Kc-1
* For each search field value ‘X’ in the sub-tree pointed at by Pi, the following condition holds: Ki-1 < X <= Ki, for 1 < I < c and, Ki-1 < X, for i = c (See diagram I for reference)
* Each internal node has at most ‘aa tree pointers.
* The root node has, at least two tree pointers, while the other internal nodes have at least \ceil(a/2) tree pointers each.
* If an internal node has ‘c’ pointers, c <= a, then it has ‘c – 1’ key values.

*Structure of  Internal Node*

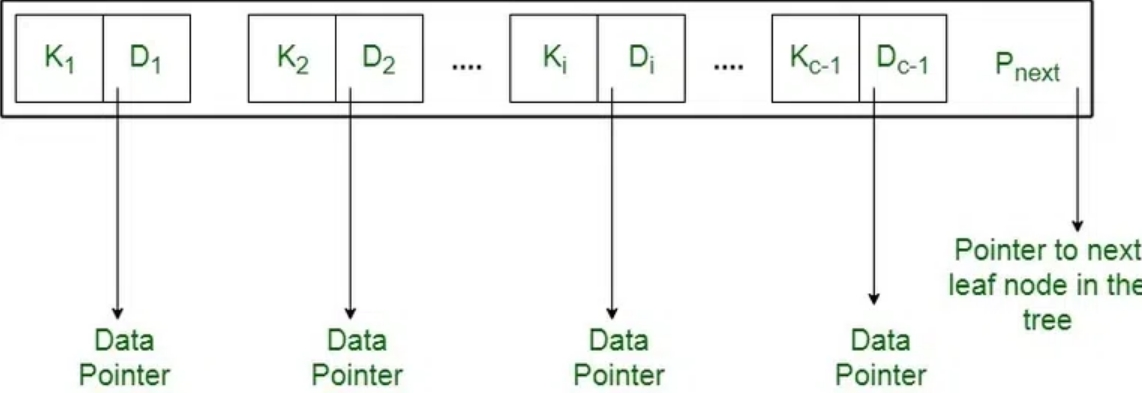
**The Structure of the Leaf Nodes of a B+ Tree of Order ‘b’ is as Follows**

* Each leaf node is of the form: <<K1, D1>, <K2, D2>, ….., <Kc-1, Dc-1>, Pnext> where c <= b and each **Di is a data pointer (i.e points to actual record in the disk whose key value is Ki or to a disk file block containing that record)** and, each **Ki is a key value** and, **Pnext points to next leaf node in the B+ tree** (see diagram II for reference).
* Every leaf node has : K1 < K2 < …. < Kc-1, c <= b
* Each leaf node has at least \ceil(b/2) values.
* All leaf nodes are at the same level.

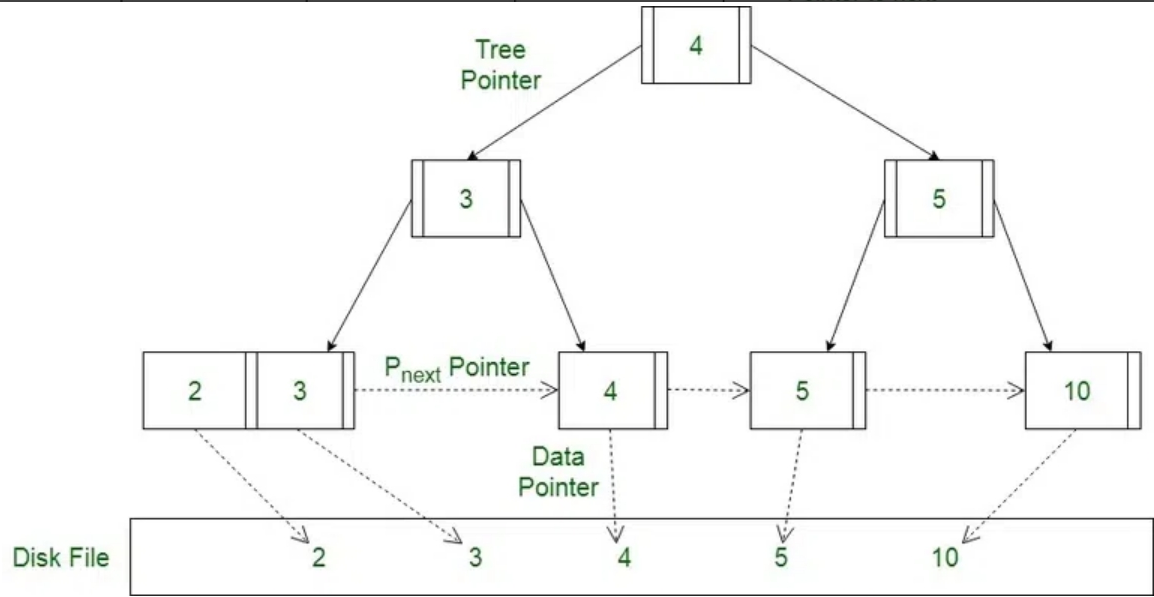


**The Structure of the Leaf Nodes of a B+ Tree of Order ‘b’ is as Follows**

* Each leaf node is of the form: <<K1, D1>, <K2, D2>, ….., <Kc-1, Dc-1>, Pnext> where c <= b and each **Di is a data pointer (i.e points to actual record in the disk whose key value is Ki or to a disk file block containing that record)** and, each **Ki is a key value** and, **Pnext points to next leaf node in the B+ tree** (see diagram II for reference).
* Every leaf node has : K1 < K2 < …. < Kc-1, c <= b
* Each leaf node has at least \ceil(b/2) values.
* All leaf nodes are at the same level.



**Diagram-II** Using the Pnext pointer it is viable to traverse all the leaf nodes, just like a linked list, thereby achieving ordered access to the records stored in the disk.



*Structure of Lead Node*