Multiway Trees

M-Way Tree

- An M-way tree is a search tree in which each node can have from **0 to m subtrees** where m=**order of the tree**
- Given a nonempty multiway tree, the following properties:
  - Each node has 0 to m subtrees
  - Given a nod with k<m subtrees, the node contains **k subtree pointers, some of which may be null** and **k-1 data entries**
- The **key values in the first subtree are all less than the key in the first entry.**

M-Way search trees

- An M-way tree is a search tree in which each node can have from 0 to m subtrees, where m= B-tree order.
- Properties of M-way tree:
  - Each node has 0 to m subtrees
  - The key values in the first subtree < key value in the first entry; key values in the other subtrees >= key value in their parent entry.
  - Keys of the data entries are ordered

  key1<= key2 <=……key n
  - All subtrees are themselves multiway trees
  - Diagram
  - Example

- **Advantages**
  - The height of the tree is greatly reduced.
- **Disadvantages**
  - it is not balanced.

Difference between binary tree and multiway tree

- Each node in a binary tree has one entry
- Each node in a multiway tree have multiple entries.

B-trees

- B-tree is an m-way search tree with the following additional properties:
  - The root is either a leaf or it has 2…m subtrees
  - All internal nodes have **atleast m/2 non-null subtrees(i.e minimum)** and **at most m non-null subtrees**
  - **A leaf node has atleast m/2 -1 non-null subtrees(i.e minimum)** and **at most m-1 non-null subtrees**

Table containing number of subtrees and no of entries for a given order

Order	Number of subtrees		Number of entries	
	Minimum(m/2)	Maximum(m)	Minimum(m/2)-1	Maximum(m-1)
3	2	3	1	2
4	2	4	1	3
5	3	5	2	4
6	3	6	2	5

- The maximum number of total nodes in a B-tree of order m and maximum height h

$$\text{Total nodes} = (m^{h+1} - 1)/(m-1)$$

- The maximum number of total keys in a B-tree of order m and maximum height h

$$\text{Total nodes} = m^{h+1} - 1$$

Inserting a value in a B-Tree

- Suppose a value k is inserted in a B-Tree. After searching for a appropriate leaf node to insert the value k, the values present at that particular leaf node is counted. This leads to 2 possibilities
  - **The leaf node is not full**
  - **The leaf node is full**
- **The leaf node is not full :**
  - **The value is inserted at its appropriate position in the node and the insertion procdure ends**

Inserting a value in a B-Tree

- **The leaf node is full**
  - **If the leaf node is full, that node is split into two nodes**

Deleting a value in a node

- Suppose a value k is deleted from a B-Tree. After searching for a appropriate leaf node to delete the value k, the values present at that particular leaf node is counted. This leads to 2 possibilities
  - **The number of values>= minimum number of values required**
  - **The number of values< minimum number of values required**

- **The number of values>= minimum number of values required**

on deleting this value , the number of values that are left in the node satisfy the condition of B-Tree.

- **The number of values< minimum number of values required**

There are two possibilities

- **The left or right sibling of the node from which the value is deleted contains more than the required minimum number of values**
  - In that case , the value of its parent is moved to the node and a value from its sibling(left or right which contains more number of value s than the required minimum values) is moved to its parent.

- **The left or right sibling of the node from which the value is deleted contains exactly the required minimum number of values**
  - The value of its parent is moved to the node and the node is merged with its sibling. If the parent also contains the minimum number of values then the same procedure of merging the node with its sibling is applied.

Difference between B-Tree and B+Tree

B Tree

- In B tree we can store both keys and data in the internal/leaf nodes.
- A B tree, on the other hand, would require a traversal of every level in the tree.
- Wastage of space

B+Tree

- In B+ tree only the data is stored in the leaf nodes.
- The leaf nodes of B+ trees are linked, so doing a full scan of all objects in a tree requires just one linear pass through all the leaf nodes.
- Used in popular file systems such as NTFS, JFS, XFS. Also using to index tables in database systems such as MYSQL
- No wastage of space

Difference between B-Tree and B+Tree

B Tree	B+Tree
• In B tree searching becomes difficult in B- tree as data cannot be found in the leaf node.	• In B+ trees, searching of any data in a B+ tree is very easy because all data is found in leaf nodes.
• They do not store redundant search key.	• They store redundant search key.

B* Trees

- The root has at least 2 and at most 2*floor[(2*m-2)/3] +1 children. Eg: for an tree of order 9 the root is full when it contains 11 children and 10 data elements.
- Non-root nodes have at most **m** and at least **ceiling[(2*m-1)/3]** children. Eg: for a tree of order 9 , a non-root node is full when there are 9 children and each non-root node must have 6 children and 5 data elements
- All paths from the root to a leaf are the same length i.e all leaves are on the same level.