Team Members:

- Aarya BC
- Sumit Dhungel
- Swapnil Tamrakar
- Pratyush Thapa

# B-Tree

A B-Tree is a self-balancing search tree. The main use of B-Trees is to minimize the frequency of disk accesses. The height of a B-Tree is kept low by adding multiple keys in a single B-Tree node. Additionally, as it is a self-balancing tree, all the leaf nodes are at the same level.

Properties of a BTree:

- A B-Tree can be definined by a minimum degree and/or a maximum degree.
- Assuming that the maximum degree is k, the minimum degree is ceiling(k/2)
- The maximum possible number of keys would be k-1 and the minimum number of keys would be (ceiling(k/2) - 1)
- Doing it the other way round, if a B-Tree is defined by a minimum degree t, the maximum degree would be 2t
- The maximum possible keys would be 2t-1 and the minimum possible keys would be t-1
- A B-Tree grows upwards. The nodes split when the node contains keys more than the maximum number of keys, where the median key becomes the root and the keys left to the median become the left child of the root and the right keys to the median become the right child of the root and so on.
- The time complexity to insert, delete and search a B-Tree is O(logn).

C++ Code Documentation:

Node

- The node is a class with a pointer to the keysArray which is an array of keys as the name suggests that stores the keys in the node.
- The minDegree is an integer value that stores the minimum degree for the BTree
- childPtrArray is an array that stores the pointer to the children of the node, based on the position of the children from the left. For example, the left child of the first element is at index 0, the right child of the first element is at index 1, the right child of the second element is at index 2 and so on.
- The numCurrentKeys stores the number of keys present in the current node
- Leaf is a Boolean value of whether or not the node is a leaf of the BTree
- It has several functions including the constructor, destructor, traverse, findKey, insert, splitchild, remove, removefromleaf, removefromnonleaf, getpredecessor,

getsuccessor, fill, borrowfromprevious, borrowfromnext, merge and so on. These are functionally broken down functions for easy accessibility and cleaner code

BTree

- The BTree is a datastructure with several nodes. So, it is a class with a node pointer root that points to the root which further points to its children and so on
- It contains a minDegree, which is an int value that stores the minimum number of nodes for the tree
- It contains several functions including the constructor, traverse, search, insert and remove(delete)

insert

- When the insert function is called, it checks if the root is NULL. At the start of the program, since the root is uninitialized, this allows for the initialization of the root.
- The new root is a Node with the first element in its keysArray as the new item to be inserted and the numCurrentKeys is incremented by 1
- If the root is not NULL, then it checks whether the root has maximum number of keys already and if so, then it splits the root and creates a new node with the median value and two children else it calls the insertNonFull function of the child node that recursively calls insertNonFull until it is in its right position.

deletekey

- It first checks if the root is empty and prints an error message if it is the case\
- Else, it calls the remove function of the root node that recursively calls remove until the item is found and then removes it. If the item is not present in the tree, it sends and error message
- Additionally, if the root itself is the lead and the number of the current keys in the root is now zero after the removal of the key, it sets the root to NULL

searchkey

- The search function returns NULL if the root is NULL
- Else, it calls the search function of the root node that recursively calls search on the child nodes until the item is found. If the item is not found, it returns NULL.

print

- This function prints the tree by traversing through it while recursively calling it from the child nodes