11/11/2020 B-Tree Insertion

Sahana L
1BM18CS089

```
void BTree::insert(int k)
{
    if (root==NULL) {
        root = new BTreeNode(t,true); //Allocate memory for root
        root->keys[0] = k;  // Insert key
        root->n = 1;} //update no of keys in root.
    else {
        if (root->n==2*t-1){// root is full; tree grows.
            BTreeNode *s=new BTreeNode(t,false); //New root
            s->C[0] = root; //old root is child of new root.
            s->splitChild(0,root);
            int i=0;
            if (s->keys[0] < k)
                i++;
            s->C[i]->insertNonFull(k);
            root = s;} // Change root
        else  // root is not full
            root->insertNonFull(k);
    }
}

void BTreeNode::insertNonFull(int k)
{
    int i=n-1; // Initialize index as index of rightmost element
    if (leaf==true) {
        while (i>=0 && keys[i]>k){
            keys[i+1]=keys[i];
            i--;}
        keys[i+1]=k;
        n=n+1;}
    else {
```

```
        while (i>=0 && keys[i] >k)
            i--;
        if (C[i+1]->n == 2*t-1){
            splitChild(i+1, C[i+1]);
            if (keys[i+1] <k)
                i++; }
        C[i+1] -> insertNonFull(k); }
    }
void BTreeNode::splitChild(int i, BTreeNode *y)
{
    BTreeNode *z = new BTreeNode(y->t, y->leaf);
    z->n = t-1;
    for (int j=0; j<t-1; j++)
        z->keys[j] = y->keys[j+t];
    if (y->leaf == false)
    {
        for (int j=0; j<t; j++)
        z-> C[j] = y->C[j+t];
    }
    y->n = t-1; // Reduce no. of keys in y.
    for (int j=n; j>=i+1; j--)
        C[j+1]= C[j];
    C[i+1]= z;
    for (int j= n-1; j>=i; j--)
        keys[j+1] = keys[j];
    keys[i] = y->keys[t-1];
    n = n+1;
}
```