ADS- Lab

B-Tree

Dhanaraj K
5A
IBM18CS027
11/11/20

→ Btree node.

```
class Node{
    int *keys;
    int t;
    Node **C;
    int n;
    bool leaf;
    Public:
    Node(int _t, bool _leaf);
    Void insertNonFull(int k);
    Void SplitChild(int i, Node* y);
    Void traverse();
    Node* findKey(int k);
    friend class BTree;
};

class BTree{
    Node *root;
    int t;
    Public:
    BTree(int _t){
        root = NULL;
        t = _t;
    }
    Void traverse(){
        if(root != NULL)
            root->traverse();
    }
    Void deletion(int k);
    Void insert(int k);
};
```

```
// inserting an element.
void BTree :: insert(int k) {
    if (root == NULL) {
        root = new Node (t, true);
        root -> keys[0] = k;
        root -> n = 1;
    } else {
        if (root -> n == 2 * t - 1) {
            Node *s = new Node (t, false);
            s -> C[0] = root;
            s -> splitChild (0, root);
            int i = 0;
            if (s -> keys[0] < k)
                i++;
            s -> C[i] -> insertNotFull(k)
            root = s;
        } else
            root -> insertNonFull (k);
    }
}
```

```
Void BTree :: 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;
    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;
}

Void BTree :: insertNonFull(int k) {
    int i = n-1;
    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) {
            Split Child (i+1, C[i+1]);
        if (Keys[i+1] < k)
            i++;
    }
    C[i+1] -> insertNonFull(k);
}
}
```