11/11/2020 ADS - LAB - 6 REVANTH. R
1BM18CS082

Insertion B-TREE

```
void Btree :: insert (int key)
{
    if (root == NULL)
    { root = new BtreeNode (t, true);
      root -> keys[0] = k;
      root -> n = 1;
    }

    if (root -> n == 2*t - 1)
    { BtreeNode *s = new BtreeNode (t, false);
      s -> C[0] = root;
      s -> splitChild(0, root);

      int i = 0; if (s -> keys[0] < k)
                    i++;
      s -> C[i] > insertNonFull(k);
      root = s;
    }
    root -> insertNonFull(k); } }
```

// Function to insert new key in this node

```
void BtreeNode :: insertNonFull (int k)
{
    int i = n-1;
    if (leaf == true)
    { while ( i >= 0 && keys[i] > k)
      { keys[i+1] = keys[i];
        i--;
      }
```

```
// Insert new key at found location
    keys[i+1] = k;
                n = n+1; }
{   // Find child which is going to have
                the new key
    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;
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];
```

```
// Copy the middle key of y to this node
    keys[i] = y->keys[t-1];
    n = n+1;
}
```