11/11/2020

ADS Lab - 7

Triveni.y
1BM19CS411

program to implement the insertion of B-tree.

```
Void BTree :: insert (int K)
{
    if (root = =NULL)
    {
        root = new BTreeNode (t, true);
        root -> KeyS[0] = K;
    }   root -> n = 1;
    else
    {
        if (root -> n == 2*t - 1)
        {
            BTreeNode *s = new BTree
                                 Node (t,
                                 false);
            s -> C[0] = root;
            s -> splitchild (0, root);

            int i=0;
            if (s->keys[0] <K)
                i++;
            s -> C[i] -> insert nonFull(K);
            root = S;
        }
        else
        {   root -> insert nonFull(K);
        }
    }
}
```

1BM19CS411 Triveni.y

```
void BTreeNode :: 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
      {
        #To find child which is going to
              have a new node

        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);
      }
```

1BM19CS411

Triveni.y

```
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;
    Keys[i] = y->keys[t-1];

    n = n+1;
}
```