11-11-2020

B Praneeth

1BM18CS023

ADS lab

B-tree insertion

```
class Node {
    int *keys; //Pointer to the array of keys in the node
    int t;     //Min number of key that must be present in a
                                                        non-root node
    Node **C; //Pointers to child nodes
    int n;     //Number of keys in the node
    bool leaf; // leaf node or not
}


void insert(int k){
    if(root == NULL){
        root = new Node(t, true);
        root -> keys[0] = k;
        root -> n = 1;
    }
    else{
    //If root node is full
    if (root -> n == 2*t -1){
```

B Praneeth

```
      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] -> insertNonFull (K);
          root = s;
       }
      else
          root -> insertNonFull(k);
   }
}

void 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 && key [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);
    }
}

void  splitChild (int i, Node * y){
     for ( j = 0; j < t-1;  j ++ )
         z → keys[j] = y → keys [j+t];
     if(y → leaf == false ){
         for (j = 0; j < t; j++){z → C[j] = y → C[j+t]; }
```

```
y->n=t-1;
 for (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 ;

}
```