Jai Jain

1BM18CS040

B - Trees

```
def B - Tree Insertion (T, K) :
        root = self. root
        if ( len (root. keys) = = (2* self. t ) - 1) :
                p = Node ( )
                root = p
                p. child. insert (0, root)
                self. split_child ( p, 0)
                self. insert_non full (temp, k)
        else :
                self. insert_non_full (root , k)

def insert_non full (self , x , k ) :
        i = len (x. keys) - 1
        if ( x. leaf ) :
                x. keys [i+1] = x. keys [i]
        while (i >= 0) and k[0] < x. keys [i]
                x . keys [i+1] = x. keys [i]
                i = 1
                x. keys [i+1] = k
        else :
                while   i >= 0 and k[0] < x. keys [i][0]
                        i - = 1
                i + = 1
        if (len (x. child . keys) = = 2(* self. t) - 1 :
                self . split_child (x , i)
                if (x[0]) > x. keys([i][0] :
                        i + = 1
        self. insert_non_full (x child[i] , k)
```

```
def  split(Node p)
     temp1  =  Node ()
     temp2  =  Node ()
     mid  =  max /2 - 1
     insert non-full (p[mid])

     while (i=0 , i< mid i++) :
           insert non-full (p[i] , temp1)
     while ( i = mid, i< max  i++ )
           insert non-full [p[i] , temp2)
     p. parent ⟶ ~~mid~~ node = temp1
     p. parent ⟶ node = temp2
```