11-11-2020

LAB-6

NIHA
1BM18CS060

Write a program to implement insertion operation on a B-Tree

1. Initialize x as root
2. while x is not leaf do following
   a) Find the child of x that's going to be traversed next. Let child be y
   b) If y is not full, change x to point y
   c) If y is full, split it & change x to point to one of two parts of y. If k is smaller than mid key in y then set x as the first part of y. Else second part of y. When we split y we move a key from y to its parent x.
3. The loop in step 2 steps when x is leaf. x must have space for 1 extra key as we have been splitting all nodes in advance. So simply insert k to x.

Pseucodou :-

```
insert ( int k)
{
     if(root ==NULL)
      {
        root = newBTreeNode (t, true) ;
        root → keys [0] = k ;
        root → n = 1 ;
     }
   else
   {    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 ;
     }
     else
      root → insertNonFull (k) ;
   }
}
```

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

```
void splitCChild ( int i, BTree Node *y )
{
   BTree Node * z = new BTreeNode (y→t, y→leaf);
     z→n = t-1;
    for ( 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[y] = 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 ( j=n-1; j>= i ; j--)
           keys[j+1] = keys[j];
      key[i] = y→keys[t-1];
      n=n+1;
}
```