1BM18CS074

B-trees. Insertion implementation

```
void BTree :: insert (int k)
{  if (root == NULL)
   { root = new BTreeNode (t, true); //memory for root
     root → keys[0] = k; //insert keys.
     root → n = 1;
   }
   else
   {  if (root → n == 2t-1)
      {
         BTreeNode *S = new BTreeNode (t, false)
         S → C[0] = root
         S → split child (0, root);
         int i = 0;
         if (S → keys[0] < k)
             i++;
         S → C[i] → insertKNonFull(k);
         root = S;
      }
      else
        root → insertNonFull (k)
   }
}


void. BTreeNode :: Insert NonFull (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 && 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>=1 ;j--)
    keys [j+1] = keys[j];
  keys[i] =y->keys[t-1];
  n=n+1;
}
```

[illegible]
11/11/2020