Sanjeev Kumar Singh
1BM18CS093
B5

11/11/20

Pg 1

ADS Lab 7

Program 6

Insertion in B-tree

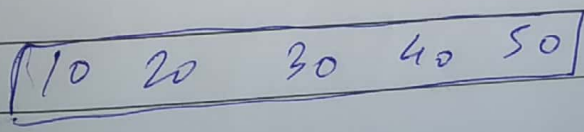
1) Initialize x as root
2) while x is not leaf, do following
   a) find the child of x that is going to be traversed next, Let child be y.
   b) If y is not full, change x to point y.
   c) If y is full, split it and change x to point to one of two parts of y. If k is smaller than mid key in y, then set x as 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 stops 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.

t = 3

e.g. Insert 10, 20, 30, 40, 50, 60, 70, 80 & 90

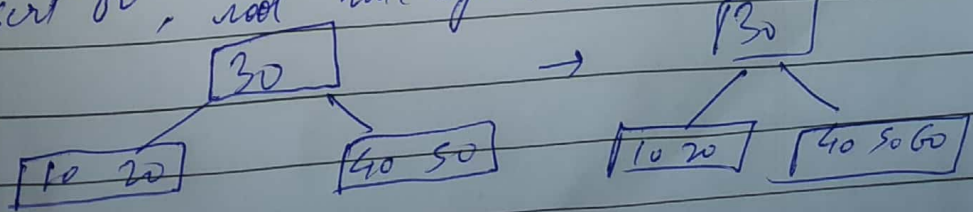
1)

2)



3) Insert 60, root node is full, hence we split into 2

i.e

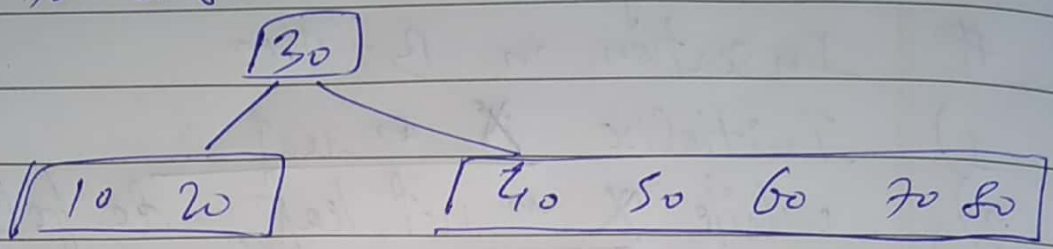


Scanned with CamScanner

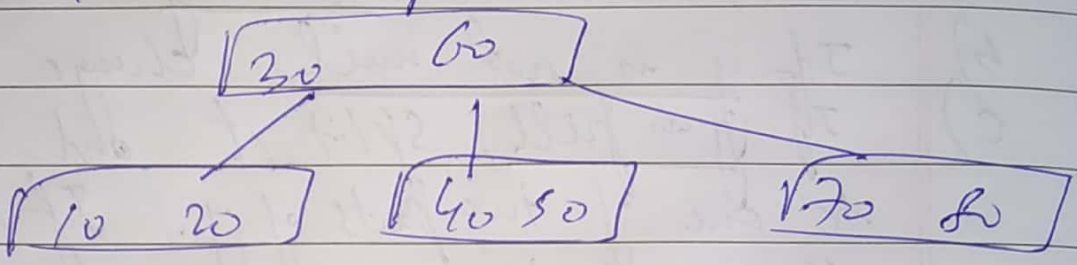
Sanjeev Kumar Singh
1BM18CS093
B5

11/10/20

4) Insert 70 & 80



Insert 90, we split



Scanned with CamScanner

Sanjeev Kumar Singh
1BM18CS093
A93

11/11/20

ADS Lab - 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 == 2t - 1) {
~~if (root → n == 2t -~~
    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);
}
}
```

Scanned with CamScanner

Sanjeev Kumar Singh
, BM18CS093
Pg 4

11/11/20

```
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++;
    }
else {
while (i >= 0 && keys[i] > k)
    i--;
if (C[i+1] -> n == 2t-1) {
    splitChild (i+1, C[i+1]);
    if (keys[i+1] < k)
        i++;
}
C[i+1] -> insertNonFull (k);
    }
}
```

Scanned with CamScanner