

Insertion function B-Tree:

```

void BTree::insert (int k) {
    if (root == NULL) {
        root = new BTreeNode(x, true);
        root->keys[0] = k;
        root->n = 1;
    }
    else if (root->n == (2 * t) - 1) {
        BTreeNode * s = new BTreeNode(x, false);
        s->c[0] = root;
        s->splitChild(0, root);
        int i = 0;
        for if (s->keys[i] < k)
            i++;
        s->c[i] -> insertNonFull(k);
        root = s;
    }
    else {
        root->insertNonFull(k);
    }
}

```

```

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 = 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->n);
 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 >= 1+i; j--)
 C[j+1] = C[j];
 C[i+1] = z;
 }
}


```
for (int j = n+1; j >= i; j--)  
    keys[i]  
    keys[j+1] = keys[j];  
    keys[i] = y → keys[t-1];  
    n++;
```

}