11/10/20

ADS - LAB

Abhishikat
1BM18CS004
5A

B-Tree Insertion

Logic

1. Initialize x as root
2. Check whether x is leaf or not, if not then do following
   - (2.1) Find the child of x that is going to be traversed next. (say, y)
   - (2.2) If y not full, change x to point to y
   - (2.3) If y is full, split it, and change x to point to one of the 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. Move key y to its parent x.
3. If x is leaf, then loop in step 2 stops. x must have space for 1 extra key ~~its parent~~ as we have been splitting all the nodes in advance. So, insert k to x.

Algorithm

```
BInsertion (T, k)
  r ← root[T]
  if n[r] = 2t - 1
     s = AllocateNode()
     root[T] = s
     leaf[s] = FALSE
     n[s] ← 0
     c1[s] ← r
     BTreeSplitChild (s, 1, r)
     BTreeInsertNonFull (s, k)
  else BTreeInsertNonFull (r, k)
BTreeInsertNonFull (r, k)
  i = n[r]
  if (leaf)[r]
     while i ≥ 1 and k < keyi[r]
        keyi+1[r] = keyi[r]
        i = i - 1
     keyi+1[r] = k
  else while i ≥ 1 and k < keyi[r]
        i = i - 1
     i = i + 1
     if n[ci[r]] == 2t - 1
        BTreeSplitChild (r, i, ci[r])
        if (k > keyi[r])
           i = i + 1
        BTreeNonFull (ci[r], k)
BTreeSplitChild (r, i)
```

for each loop

```
BTreeSplitChild(x,i,y)
z = AllocateNode()
leaf[z] = leaf[y]
n[z] = t-1
for j = 1 to t-1
    keyj[z] = keyj+t[y]
if not leaf[y]
    for j = 1 to t
        cj[z] = cj+t[y]
n[y] = t-1
for j = n[x]+1 to i+1
    cj+1[x] = cj[x]
ci+1[x] = z
for j = n[x] to i
    keyj+1[x] = keyj[x]
keyi[x] = keyt[y]
n[x] = n[x]+1
```