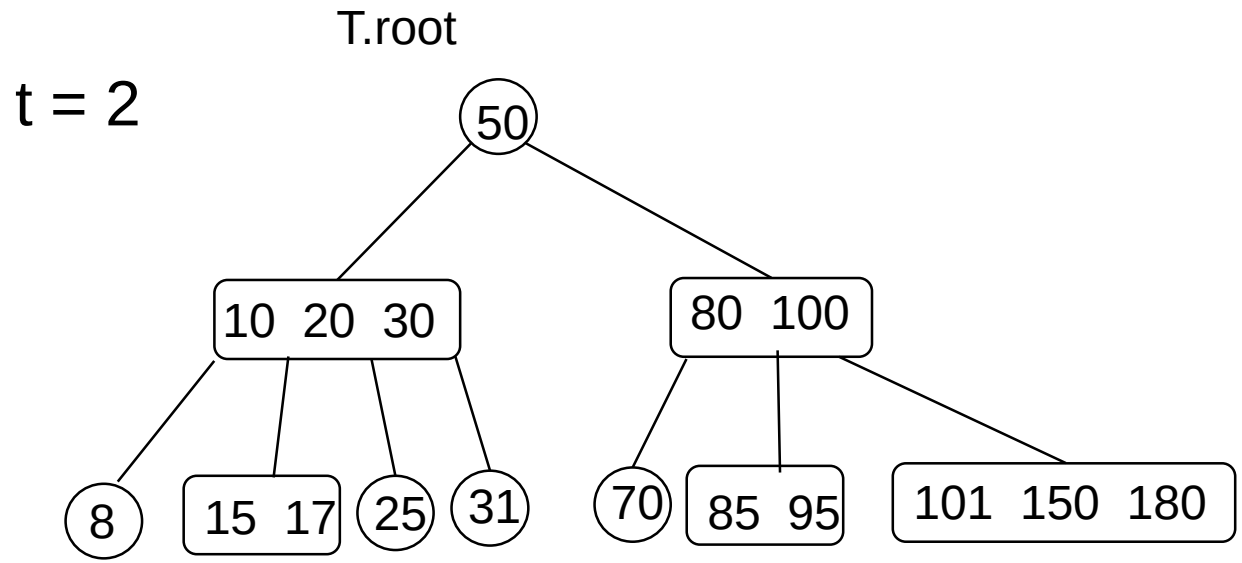
COT 6405
ANALYSIS OF ALGORITHMS

B-Trees

Computer & Electrical Engineering and Computer Science Department
Florida Atlantic University

Insert operation - strategy

- Search for a leaf where to insert the new key
- Insert into an existing leaf node
  - cannot create a new leaf
- If the leaf node is full, then split around the median key



Insert operation

- Goal: insert the key in the B-tree in a single pass from the root to a leaf
  - As the algorithm travels down the tree, it splits each full node along the way, including the leaf

Insert operation - main functions

- B-TREE-SPLIT-CHILD(x,i)
- B-TREE-INSERT(T,k)
- B-TREE-INSERT-NONFULL(x,k)

Splitting a node in a B-tree

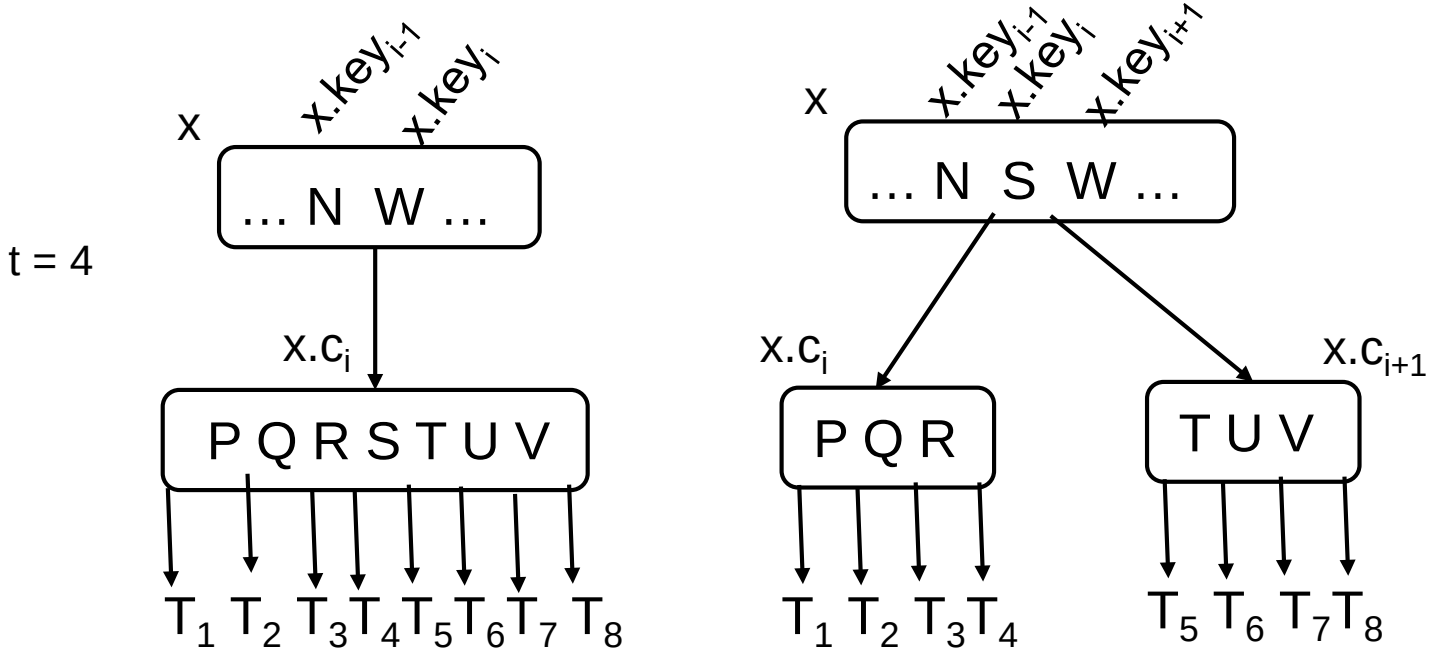
B-TREE-SPLIT-CHILD(x,i)

- Input:

  x – nonfull internal node (in the main memory)

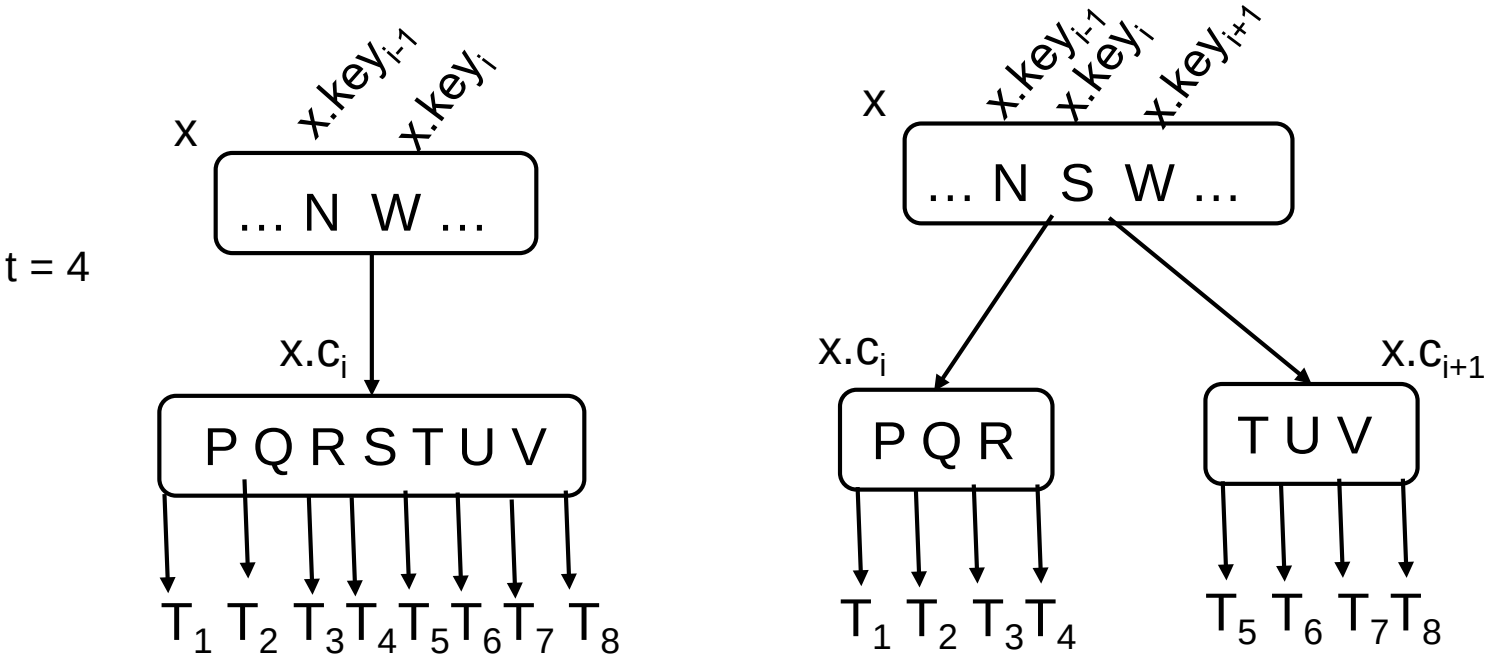
  index i such that $\begin{cases} x.c_i \text{ is a full child of } x \\ x.c_i \text{ is in the main memory} \end{cases}$

- Output: split the node $x.c_i$ around its median key $x.key_t$



B-TREE-SPLIT-CHILD(x, i)
z = ALLOCATE-NODE()
y = $x.c_i$
z.leaf = y.leaf
z.n = t – 1
for j = 1 to t –1
    $z.key_j = y.key_{j+t}$
if not y.leaf
    for j = 1 to t
        $z.c_j = y.c_{j+t}$
y.n = t – 1
for j = x.n+1 **downto** i+1
    $x.c_{j+1} = x.c_j$
$x.c_{i+1}$ = z
for j = x.n **downto** i
    $x.key_{j+1} = x.key_j$
$x.key_i = y.key_t$
x.n = x.n + 1
DISK-WRITE(y)
DISK-WRITE(z)
DISK-WRITE(x)

B-TREE-SPLIT-CHILD

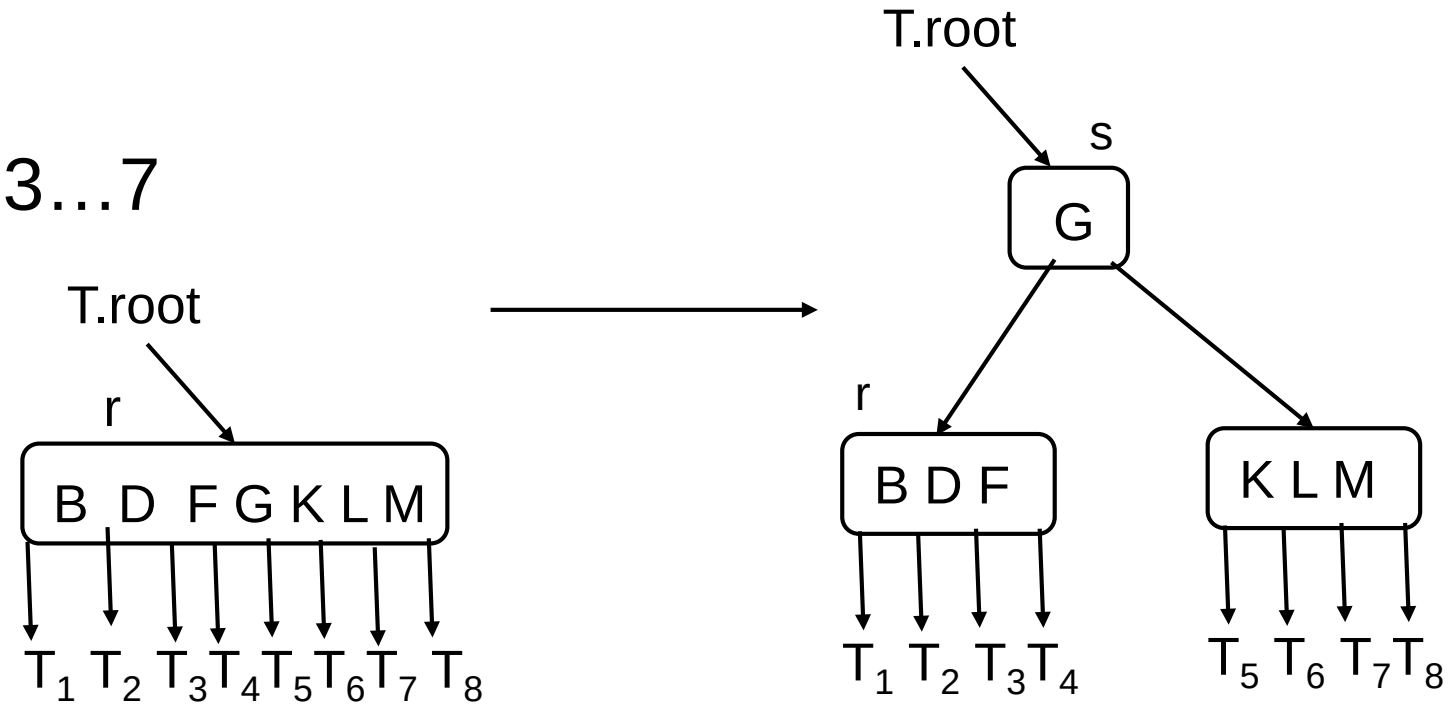


RT = Θ(t)
Θ(1) disk operations

B-TREE-INSERT()

t = 4

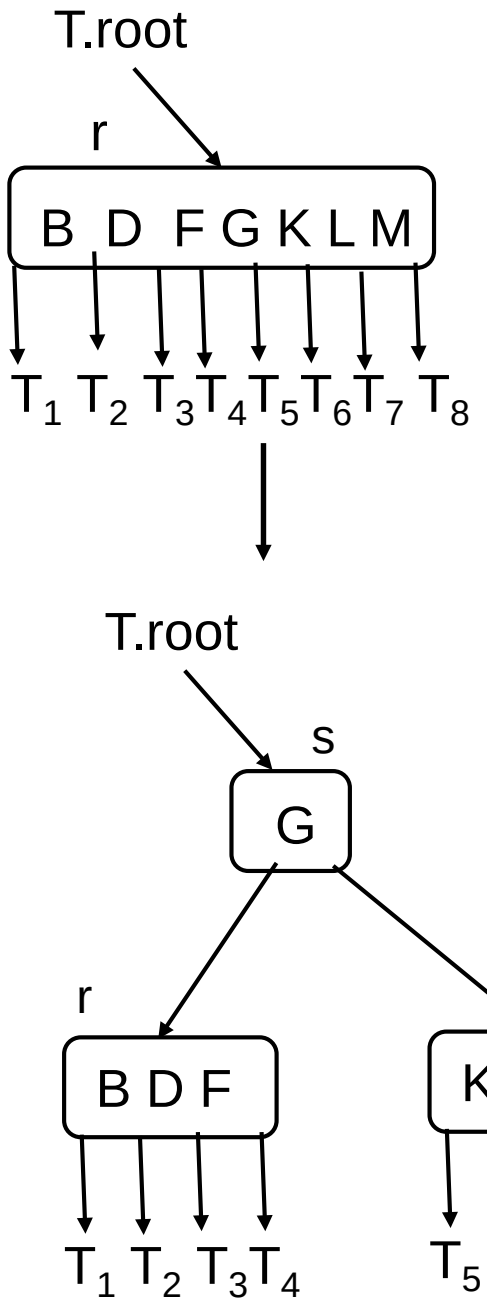
keys 3…7



- If the root r is full, then split r and a new node s becomes the root

```
B-TREE-INSERT(T, k)
r = T.root
if r.n == 2t – 1
     s = ALLOCATE-NODE()
     T.root = s
      s.leaf = FALSE
      s.n = 0
      s.c1 = r
      B-TREE-SPLIT-CHILD(s, 1)
      B-TREE-INSERT-NONFULL(s, k)
else B-TREE-INSERT-NONFULL(r,k)
```

t = 4
# keys
3…7

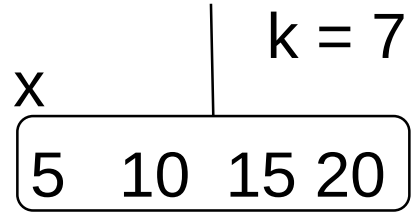


B-TREE-INSERT-NONFULL(x,k) – inserts key k into the subtree rooted at the **nonfull** node x

- node x must be nonfull when the procedure is called !

```
B-TREE-INSERT-NONFULL(x, k)
i = x.n
if x.leaf
    while i ≥ 1 and k < x.key_i
        x.key_{i+1} = x.key_i
        i = i – 1
    x.key_{i+1} = k
    x.n = x.n + 1
    DISK-WRITE(x)
else
    while i ≥ 1 and k < x.key_i
        i = i – 1
    i = i + 1
    DISK-READ(x.c_i)
    if x.c_i.n == 2t – 1
        B-TREE-SPLIT-CHILD(x, i)
        if k > x.key_i
            i = i + 1
    B-TREE-INSERT-NONFULL(x.c_i, k)
```

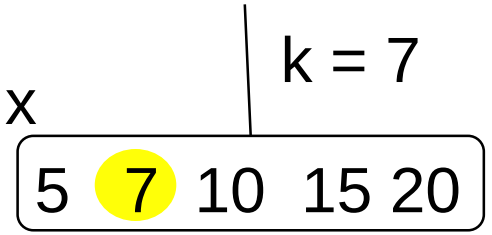
x is a leaf:



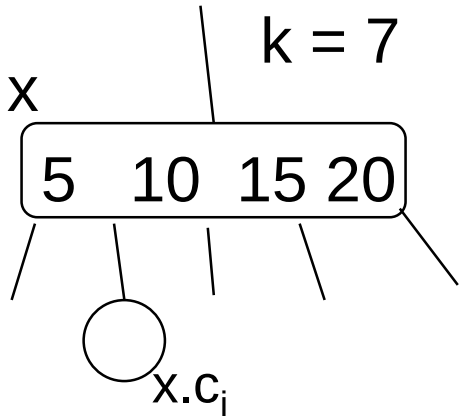
i = 4

….

i = 1



x is NOT a leaf:



RT for the insert operation

RT = $O(th) = O(t\log_t n)$

O(h) disk access operations