Design and Analysis of Algorithms

Lecture-19

Dharmendra Kumar (Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research,
Prayagraj

B-Tree creation

Example: Create B-tree for the following elements with minimum degree t = 2.

F, S, Q, K, C, L, H, T, V, W, M, R, N, P, A, B, X, Y, D, Z, E

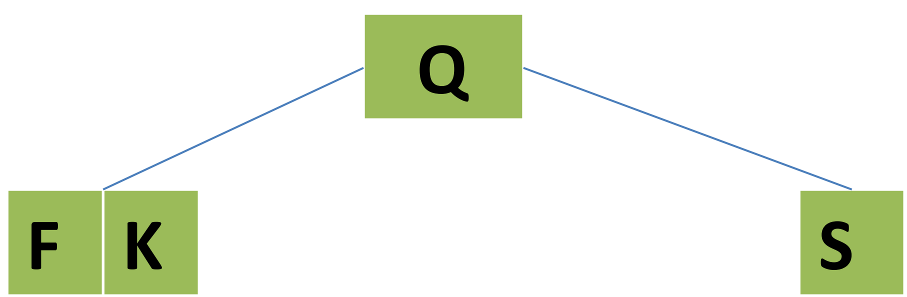
Solution:

Minimum number of keys in a node = t-1 = 1

Maximum number of keys in a node = 2t-1 = 3

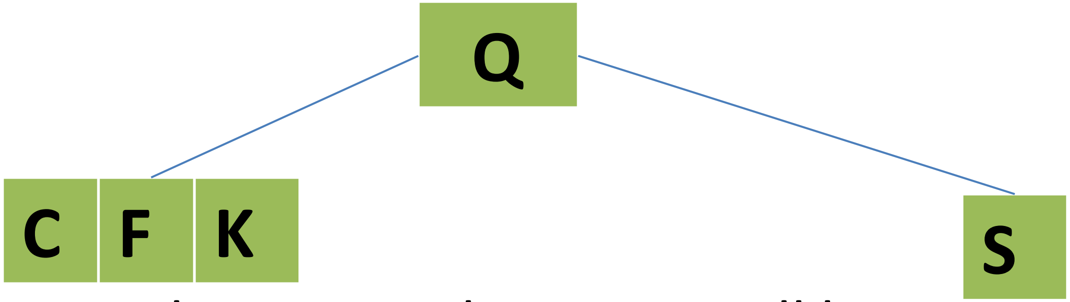
Initial consider 2t-1 elements in the sequence i.e. 3 elements. These are F, S and Q. B-tree for this will be

After inserting next element K, the B-tree will be

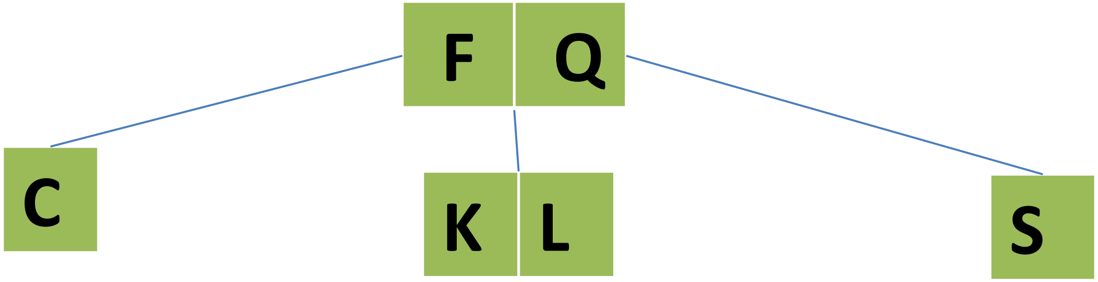


B-Tree

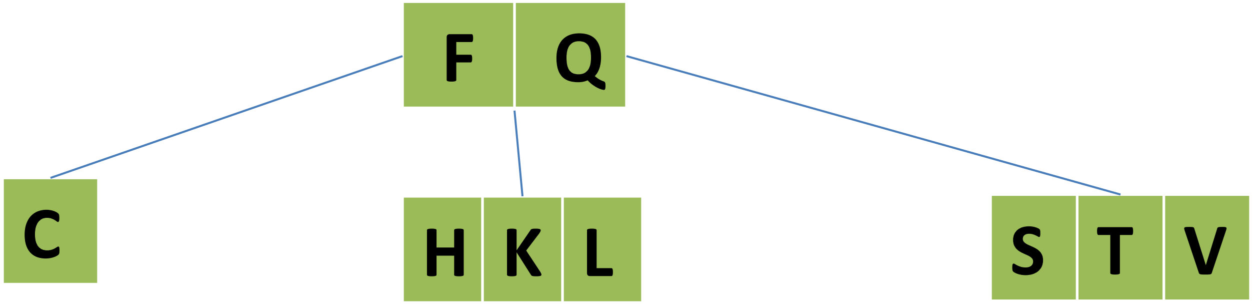
After inserting next element C, the B-tree will be



After inserting next element L , the B-tree will be

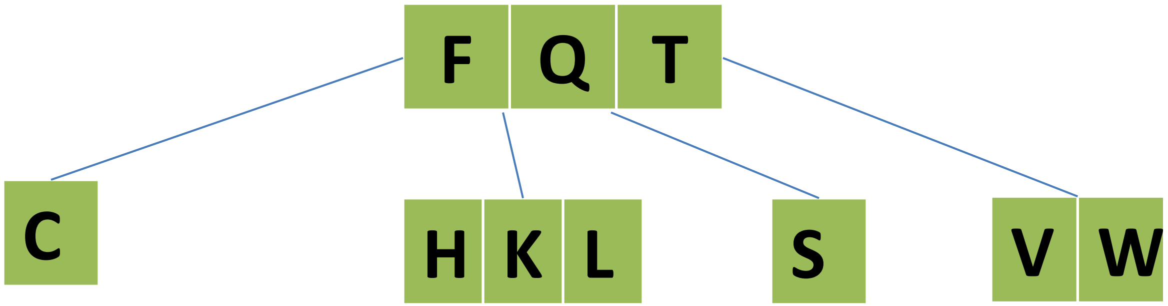


After inserting next element H,T, and V, the B-tree will be

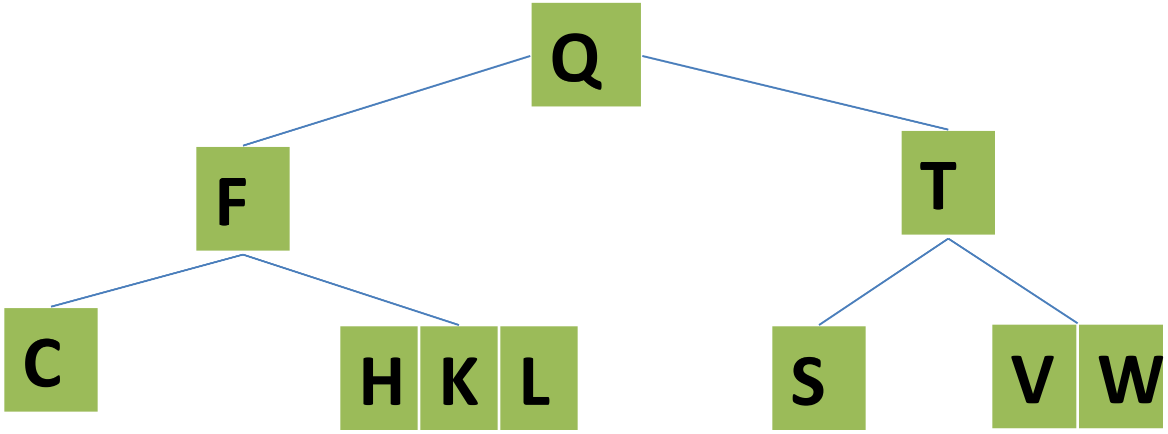


B-Tree

After inserting next element W, the B-tree will be

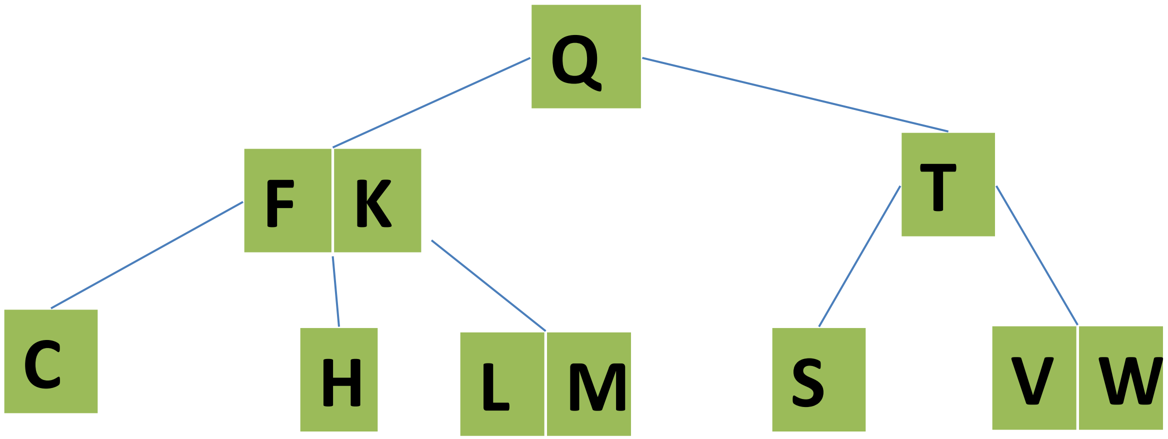


Now, insert element M. Since root node is full, therefore first, we split it.

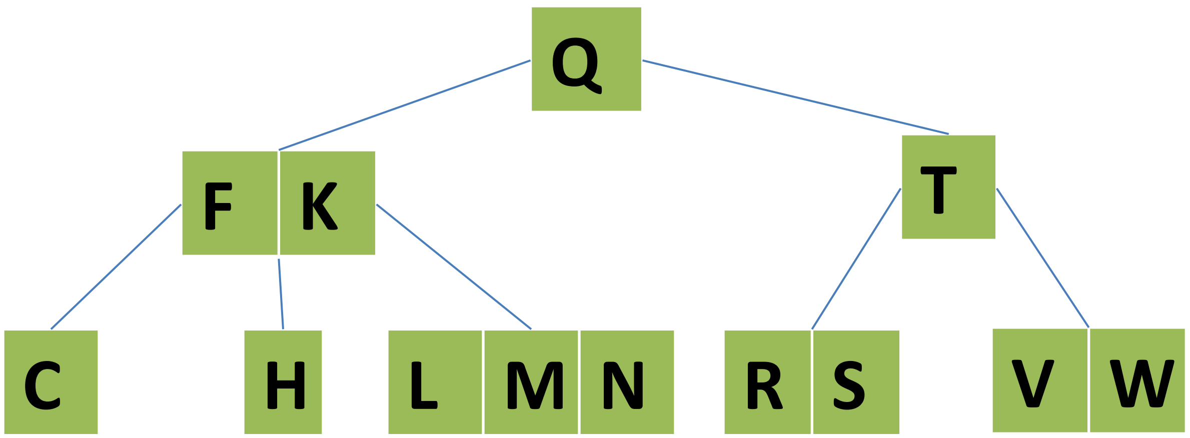


B-Tree

. After splitting and inserting M, B-tree will be

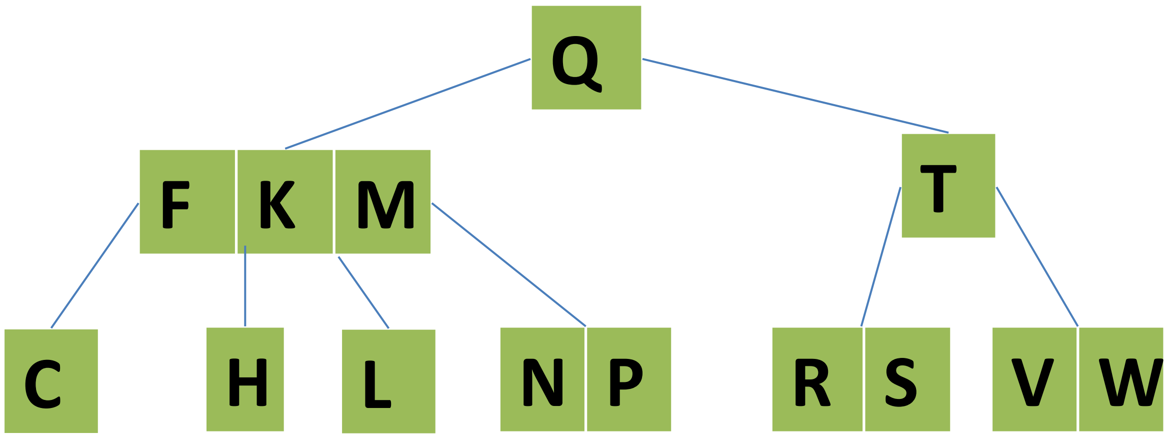


After inserting next element R and N, the B-tree will be

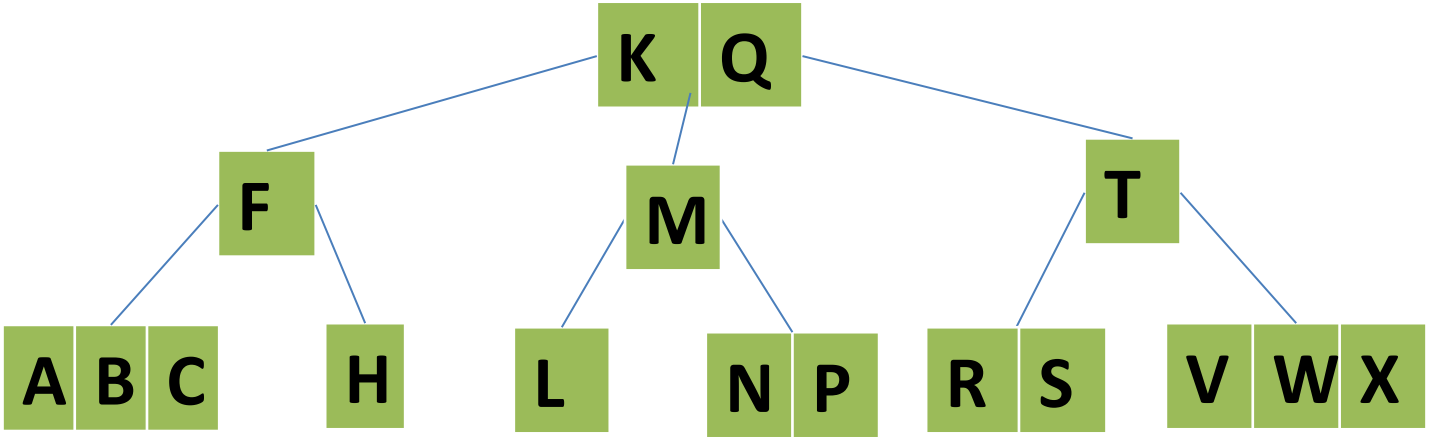


B-Tree

After inserting next element P, the B-tree will be

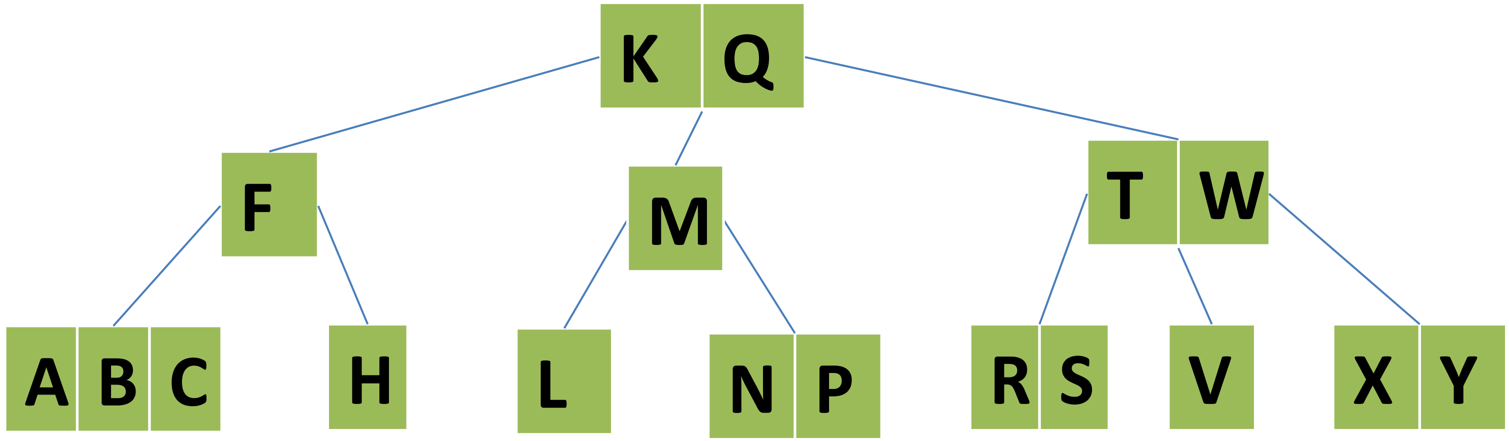


After splitting and inserting next element A, B and X, the B-tree will be

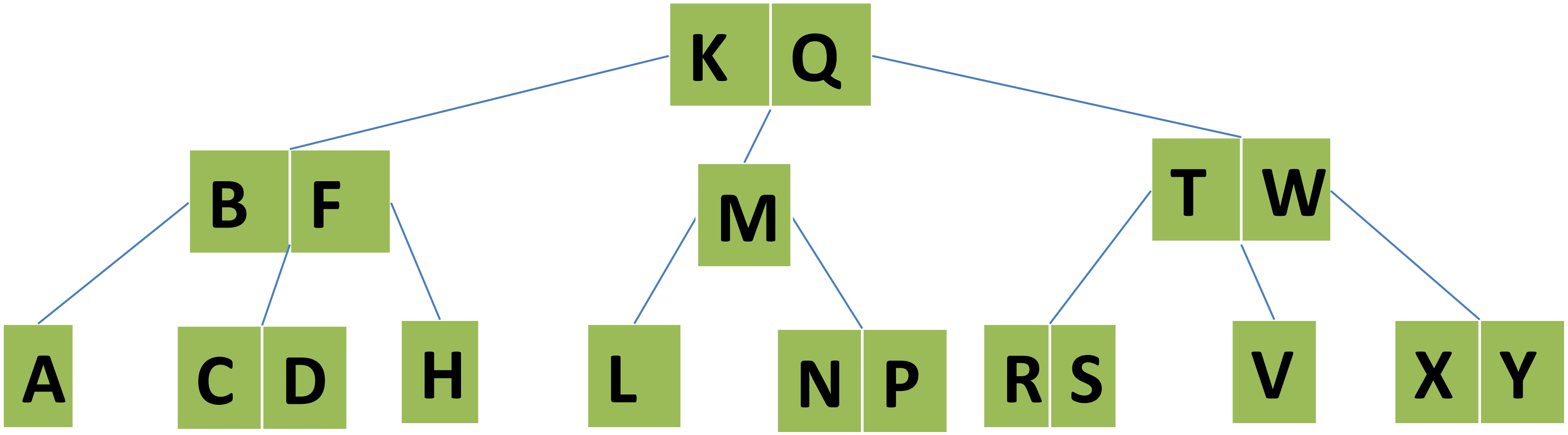


B-Tree

After splitting and inserting next element Y, the B-tree will be

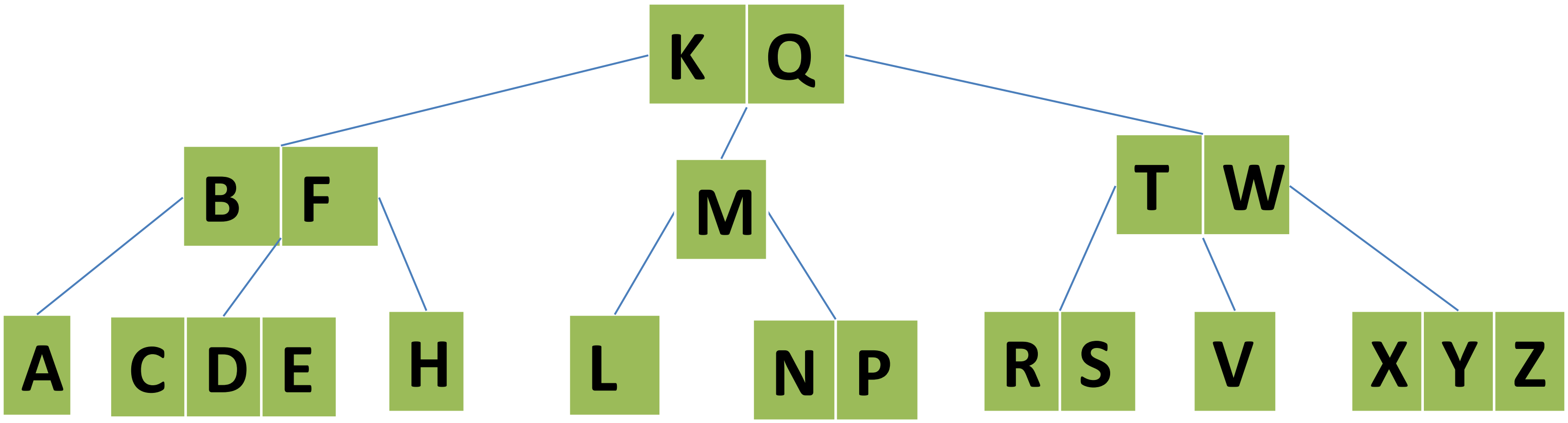


After splitting and inserting next element D, the B-tree will be



B-Tree

After inserting next element Z and E, the B-tree will be



Final B-Tree

B-Tree

Example: Create B-tree for the following elements with minimum degree t = 3.

F, S, Q, K, C, L, H, T, V, W, M, R, N, P, A, B, X, Y, D, Z, E, G, I.

Example: Insert the following keys in a *2-3-4 B Tree*:

40, 35, 22, 90, 12, 45, 58, 78, 67, 60

Example: Using minimum degree ‘t’ as 3, insert following sequence of integers

10, 25, 20, 35, 30, 55, 40, 45, 50, 55, 60, 75, 70, 65, 80, 85 and 90

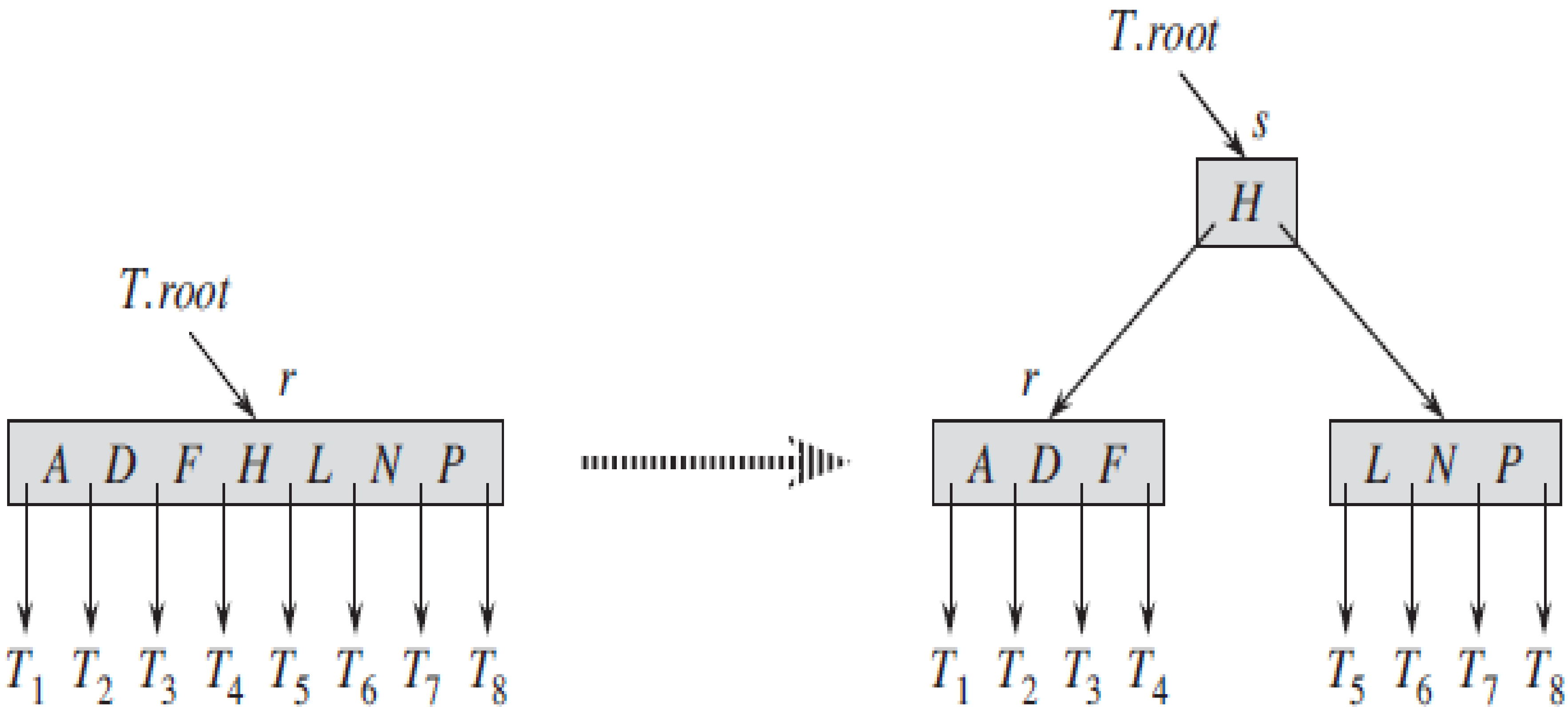
in an initially empty B-Tree. Give the number of nodes splitting operations that take place.

B-Tree Insertion Algorithm

```
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.c_1 = r
        B-TREE-SPLIT-CHILD(s, 1)
        B-TREE-INSERT-NONFULL(s, k)
else B-TREE-INSERT-NONFULL(r, k)
```

B-Tree Insertion Algorithm

Splitting the root with t = 4. Root node r splits in two, and a new root node s is created. The new root contains the median key of r and has the two halves of r as children. The B-tree grows in height by one when the root is split.

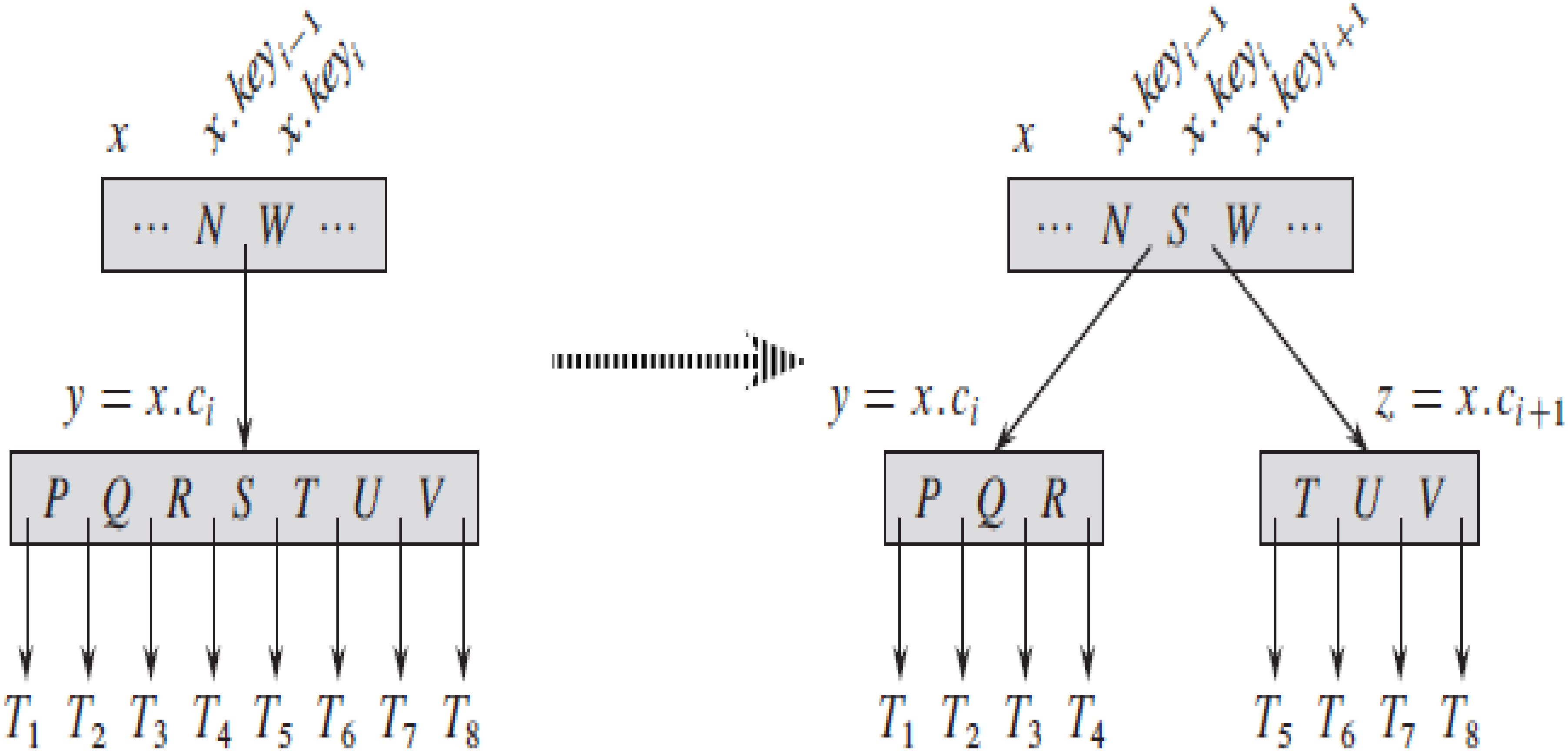


B-Tree Insertion Algorithm

```
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 Insertion Algorithm

Splitting a node with t = 4. Node y = x.c_i splits into two nodes, y and z, and the median key S of y moves up into y's parent.



B-Tree Insertion Algorithm

```
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)
```

Time complexity of insertion algorithm

$= O(th)$

$= O(t \log_t n)$