Design and Analysis of Algorithms

Lecture-18

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

B-Tree

B-Tree

Definition

A B-tree T is a rooted tree having the following properties:

1. Every node x has the following attributes:

   a. x.n, the number of keys currently stored in node x,

   b. the x.n keys themselves, $x.key_1$, $x.key_2$, ……,$x.key_{x.n}$, stored in non-decreasing order, so that $x.key_1 \le x.key_2 \le \ldots\ldots\ldots \le x.key_{x.\,n}$,

   c. x.leaf , a boolean value that is TRUE if x is a leaf and FALSE if x is an internal node.

2. Each internal node x also contains x.n+1 pointers $x.c_1$; $x.c_2$, ……. ,$x.c_{x.\,n+1}$ to its children. Leaf nodes have no children, and so their c_i attributes are undefined.

3. The keys $x.key_i$ separate the ranges of keys stored in each subtree: if k_i is any key stored in the subtree with root $x.c_i$, then

$k_1 \le x.key_1 \le k_2 \le x.key_2 \le \ldots\ldots\ldots\ldots\ldots\ldots \le X.key_{x.\,n} \le k_{x.\,n+1}$.

B-Tree

4. All leaves have the same depth, which is the tree's height h.

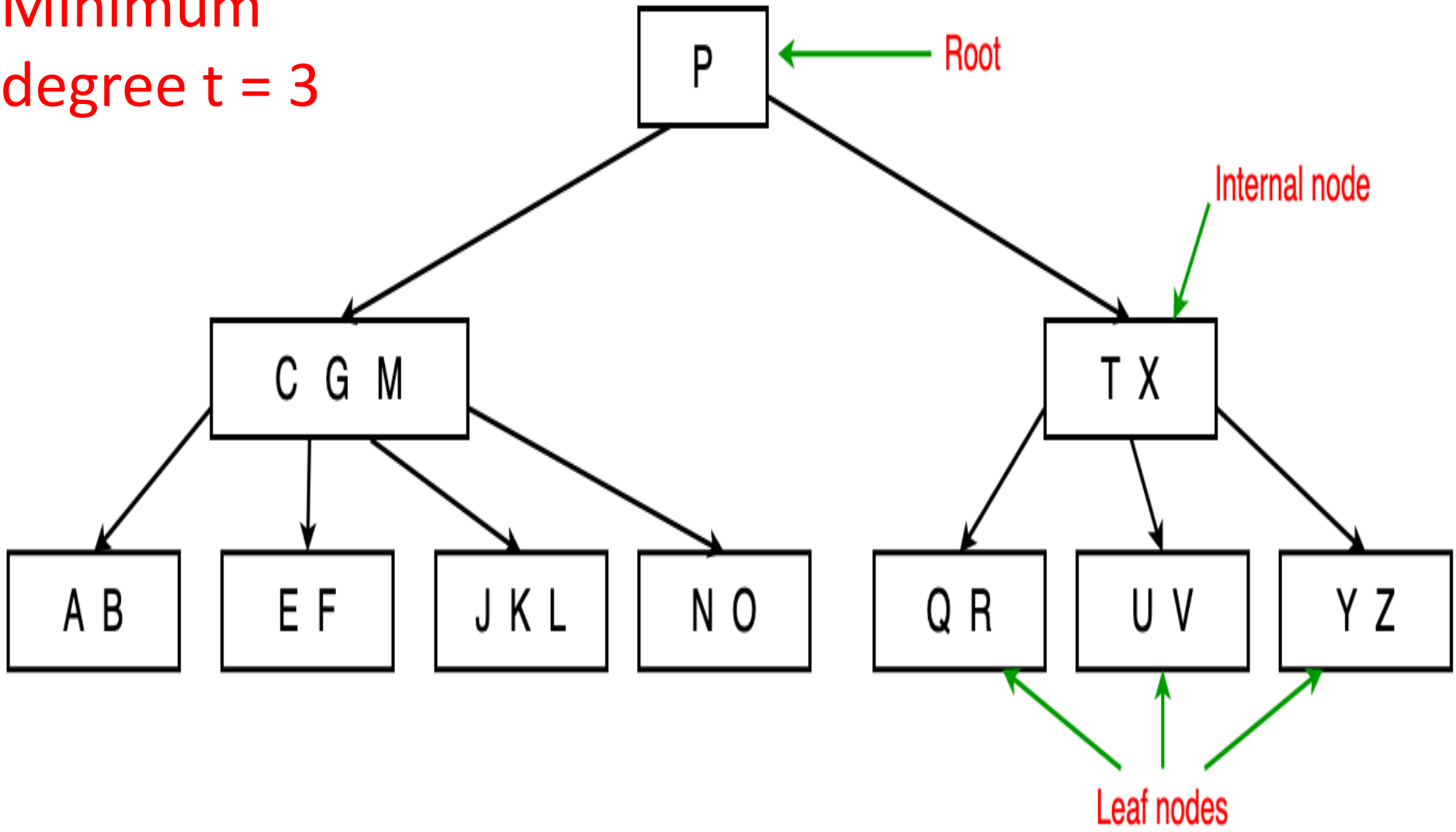
5. Nodes have lower and upper bounds on the number of keys they can contain. We express these bounds in terms of a fixed integer $t \geq 2$ called the minimum degree of the B-tree:

    a. Every node other than the root must have at least t-1 keys. Every internal node other than the root thus has at least t children. If the tree is nonempty, the root must have at least one key.

    b. Every node may contain at most 2t -1 keys. Therefore, an internal node may have at most 2t children. We say that a node is full if it contains exactly2t -1 keys.

B-Tree

Minimum degree t = 3



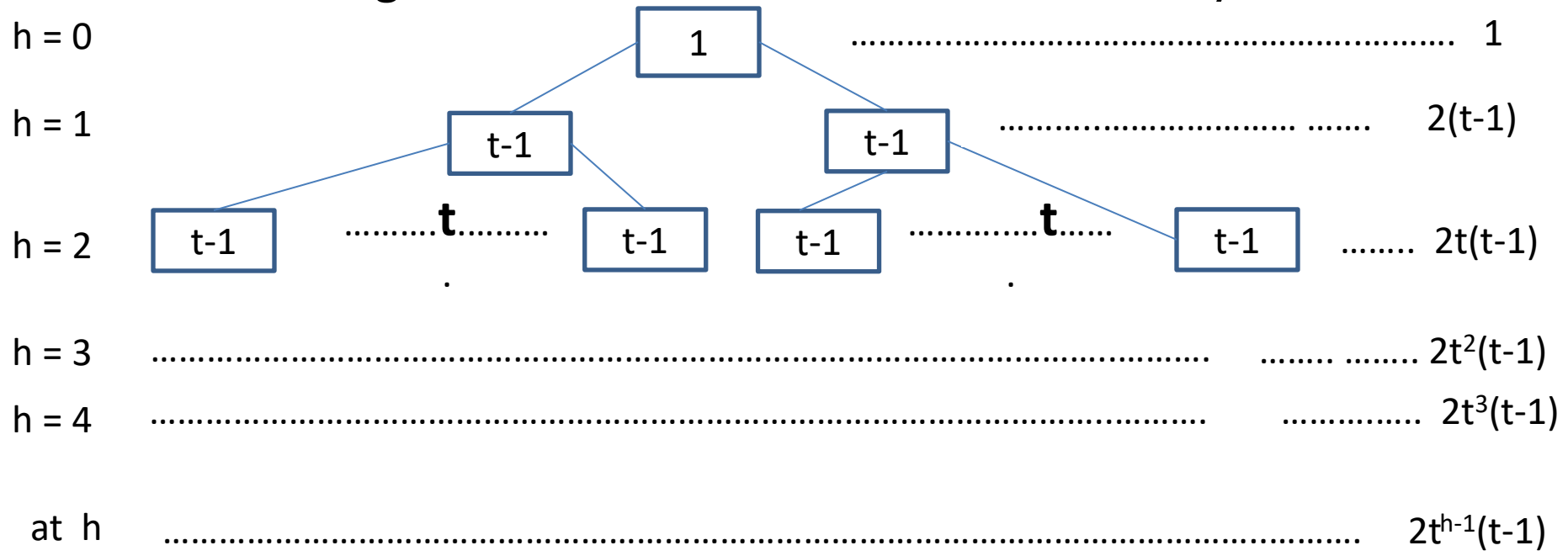
B-Tree

Theorem: If n ≥ 1, then for any n-key B-tree T of height h and minimum degree t ≥ 2,

$$h \leq \log_t \frac{(n+1)}{2}$$

Proof: To prove this, first we will find the minimum number of keys in the B-tree with minimum degree t and height h.

Consider following structure of B-tree with minimum keys:-



B-Tree

Therefore, the minimum number of the keys in B-tree with minimum degree t and height h

$$= 1+ 2(t-1) + 2t(t-1) + 2t^2(t-1) + 2t^3(t-1) + \ldots\ldots+2t^{h-1}(t-1)$$

$$= 1 + 2(t-1)\ (1 + t + t^2 + t^3 + \ldots\ldots\ldots + t^{h-1})$$

$$= 1 + 2(t-1)\ \frac{(th-1)}{(t-1)}$$

$$= 1 + 2(t^h-1)$$

$$= 1 + 2t^h-2$$

$$= 2t^h-1$$

Now, since the number of keys in given B-tree is n, therefore

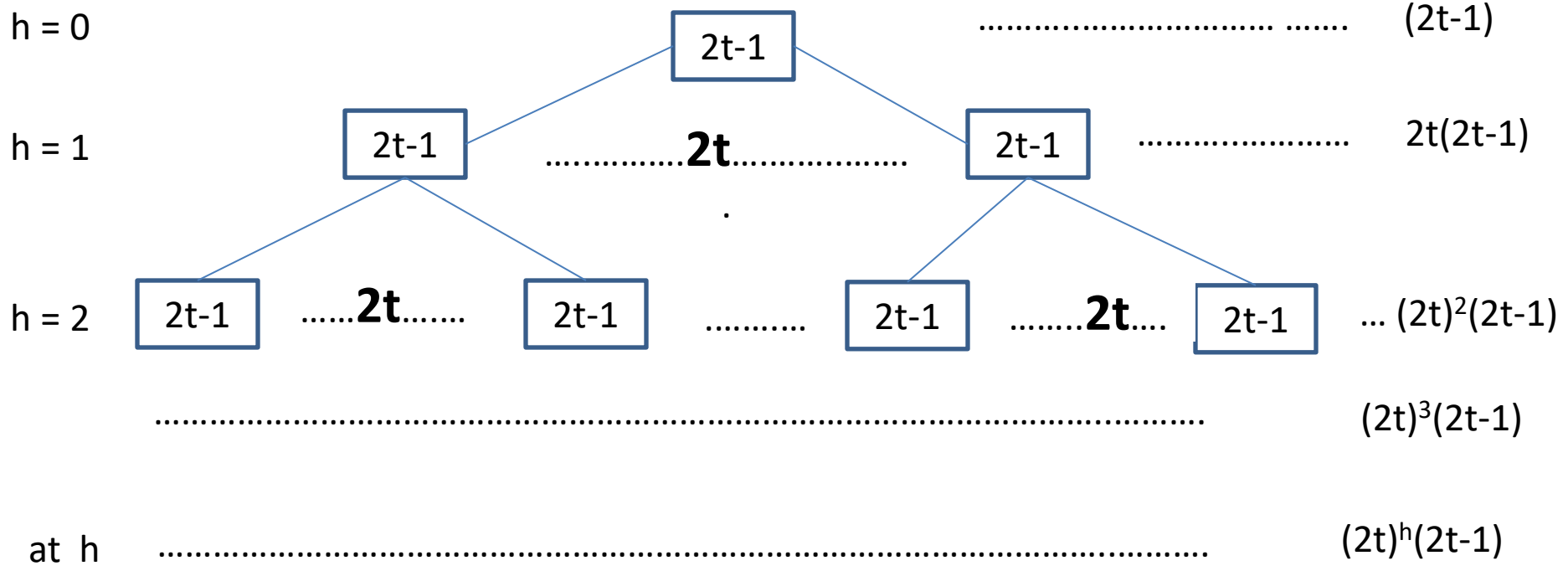
$$2t^h-1 \leq n \ \Rightarrow\ t^h \leq \frac{(n+1)}{2} \Rightarrow h \leq \log_t \frac{(n+1)}{2}$$

It is proved.

B-Tree

Question: As a function of the minimum degree t , what is the maximum number of keys that can be stored in a B-tree of height h?

Solution: Consider following structure of B-tree with maximum keys:-



B-Tree

Therefore maximum number of keys in B-tree with minimum degree t and height h

$= (2t-1) + 2t(2t-1) + (2t)^2(2t-1) + (2t)^3(2t-1) + \ldots\ldots\ldots\ldots + (2t)^h(2t-1)$

$= (2t-1)(1+2t + (2t)^2 + (2t)^3 + \ldots\ldots\ldots\ldots\ldots\ldots + (2t)^h)$

$= (2t-1)\frac{((2t)^{h+1}-1)}{(2t-1)}$

$= (2t)^{h+1}-1$

B-Tree

Note:

The simplest B-tree occurs when t = 2. Every internal node then has either 2, 3, or 4 children, and we have a 2-3-4 tree. In practice, however, much larger values of t yield B-trees with smaller height.

B-Tree of order m

B-tree of order m is a tree which satisfies the following properties:

1. Every node has at most m children.
2. Every non-leaf node (except root) has at least ⌈m/2⌉ child nodes.
3. The root has at least two children if it is not a leaf node.
4. A non-leaf node with k children contains k − 1 keys.
5. All leaves appear in the same level and carry no information.

Each internal node's keys act as separation values which divide its subtrees. For example, if an internal node has 3 child nodes (or subtrees) then it must have 2 keys: a1 and a2. All values in the leftmost subtree will be less than a1, all values in the middle subtree will be between a1 and a2, and all values in the rightmost subtree will be greater than a2.

Searching operation in B-Tree

Searching a B-tree is much like searching a binary search tree, except that instead of making a binary, or "two-way," branching decision at each node, we make a multiway branching decision according to the number of the node's children.

```
B-TREE-SEARCH(x, k)
1  i = 1
2  while i ≤ x.n and k > x.key_i
3      i = i + 1
4  if i ≤ x.n and k == x.key_i
5      return (x, i)
6  elseif x.leaf
7      return NIL
8  else DISK-READ(x.c_i)
9      return B-TREE-SEARCH(x.c_i, k)
```

Searching operation in B-Tree

- B-TREE-SEARCH takes as input a pointer to the root node x of a subtree and a key k to be searched for in that subtree.
- The initial call of this algorithm will be

  B-TREE-SEARCH(T.root, k)
- If k is in the B-tree, B-TREE-SEARCH returns the ordered pair (y, i) consisting of a node y and an index i such that $y.key_i = k$. Otherwise, the procedure returns NIL.