B-Tree

1

B-Tree

- Original B-Tree proposed by R. Bayer and E. McCreigh in 1972.
- A B-Tree is a specialized multi-way tree designed especially for use on external disk.
- Improved versions of B-Trees were later proposed in 1982 by Huddleston and Mehlhorn, and by Maier and Salveter.
- B-tree variants are used mostly today as index structures in database applications.

2

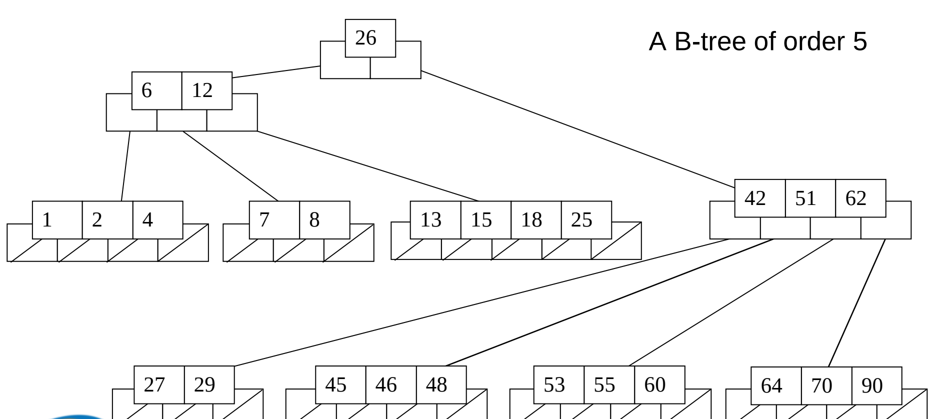
fit@hcmus

Definition

- *(Knuth's definition)* A B-tree of order *m* is a tree which satisfies the following properties:
  - Every node has at most *m* children.
  - Every non-leaf node (except root) has at least $\lceil m/2 \rceil$ child nodes.
  - The root has **at least two children** if it is not a leaf node.
  - A non-leaf node with *k* children contains $\boldsymbol{k-1}$ keys.
  - All leaves appear in the same level and carry no information.
- The number *m* should be (always) odd.

fit@hcmus

Example

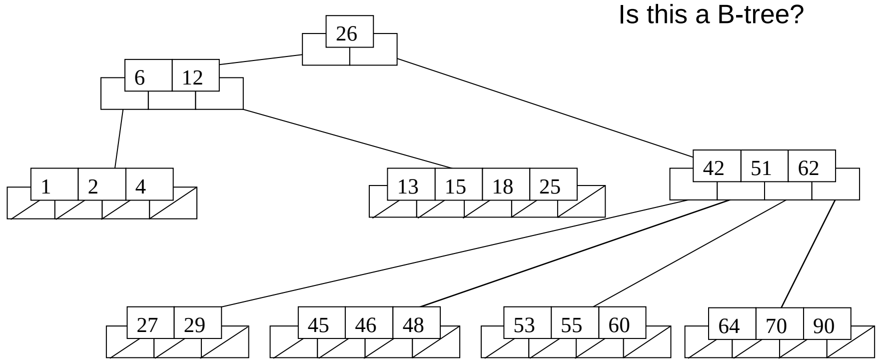


A B-tree of order 5

fit@hcmus

Example

Is this a B-tree?



7

fit@hcmus

Height of B-Tree

- The maximum number of keys in a B-tree of order m and height h:

root	$m - 1$
level 2	$m(m - 1)$
level 3	$m^2(m - 1)$
. . .	
level h	$m^{h-1}(m - 1)$

- So, the total number of keys is

$$(1 + m + m^2 + m^3 + \ldots + m^{h-1})(m - 1) =$$
$$[(m^h - 1)/ (m - 1)]\ (m - 1) = \mathbf{m^h - 1}$$

8

fit@hcmus

Operations

- Insert (a key)
- Remove (a key)

fit@hcmus

Insertion

Insertion

- B-tree insertion is a generalization of 2-3 tree insertion.
- Insert K into B-tree of order m.
  - We find the insertion point (in a leaf) by doing a search.
  - If there is room then insert K.
  - Else, promote the middle key to the parent, split the node into nodes around the middle key.
- If the splitting backs up to the root, then
  - Make a new root containing the middle key.
- Note
  - The tree grows from the leaves, balance is always maintained.

Example

- Perform step-by-step the insertion of the following values to the initially empty B-tree order 5

1, 12, 8, 2, 25, 5, 14, 28, 17

1 **12** **8** **2** 25 5 14 28 17

fit@hcmus

Example

1	2	8	12

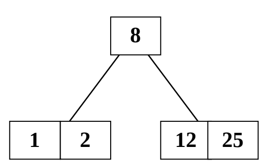
Insert **1** Insert **12** Insert **8** Insert **2**

1 12 8 2 **25** 5 14 28 17

fit@hcmus

Example

1	2	8	12	25

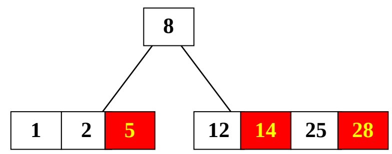


Insert **25**

1 12 8 2 25 **5** **14** **28** 17

fit@hcmus

Example

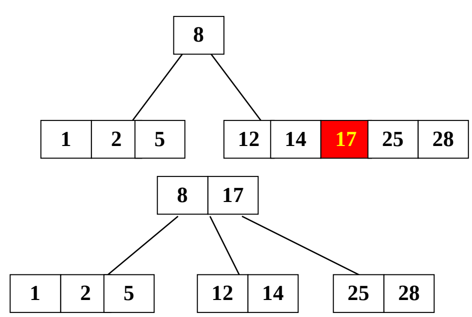


Insert **5, 14, 28**

1 12 8 2 25 5 14 28 **17**

fit@hcmus

Example



Insert **17**

Deletion

Deletion

- If the key to be deleted is not in a leaf, swap it with its successor (or predecessor). Then delete the key from the leaf.
- If leaf contains more than the minimum number of keys, then one can be deleted with no further action.

fit@hcmus

Deletion

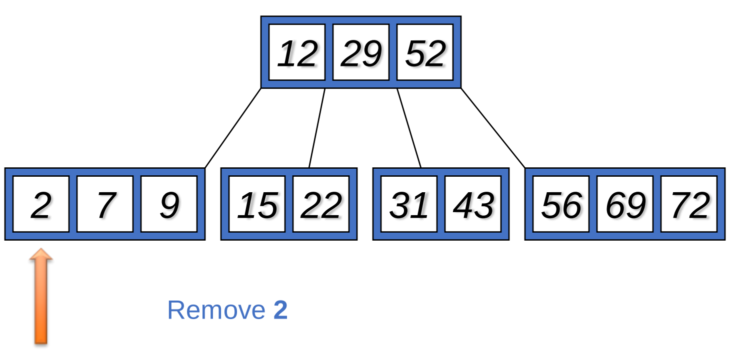
- If the node contains the minimum number of keys, consider the two immediate siblings of the parent node:
  - If one of these siblings has more than the minimum number of keys, then **redistribute** one key from this sibling to the parent node, and one key from the parent to the deficient node.
  - If both immediate siblings have exactly the minimum number of keys, then **merge** the deficient node with one of the immediate sibling node and one key from the parent node.

- If this leaves the parent node with too few keys, then the process is propagated upward.

fit@hcmus

Example

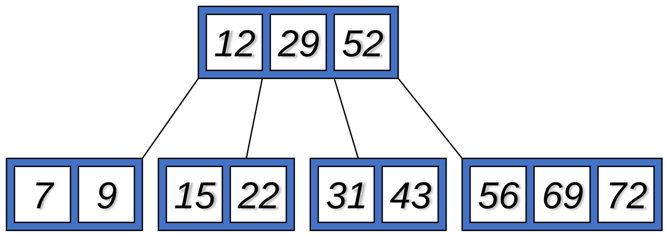
Given a B-tree order 5:



Example

fit@hcmus

Given a B-tree order 5:



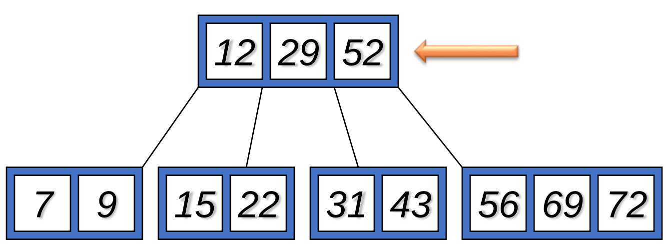
After removing **2**

21

Example

fit@hcmus

Given a B-tree order 5:



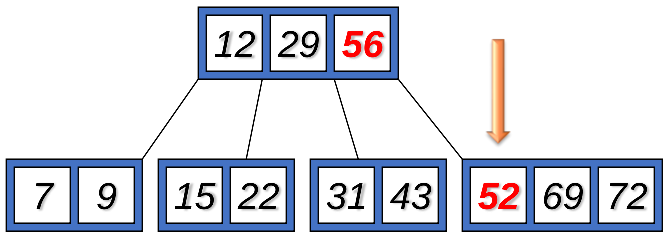
Remove **52**

22

fit@hcmus

Example

Given a B-tree order 5:

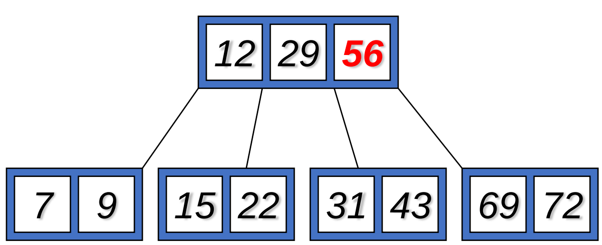


Remove **52** (replaced by **56**)

fit@hcmus

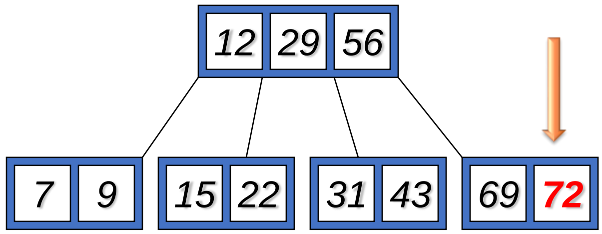
Example

Given a B-tree order 5:



Example

Given a B-tree order 5:

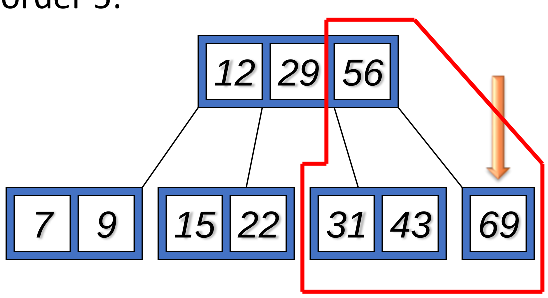


Remove **72**

25

Example

Given a B-tree order 5:

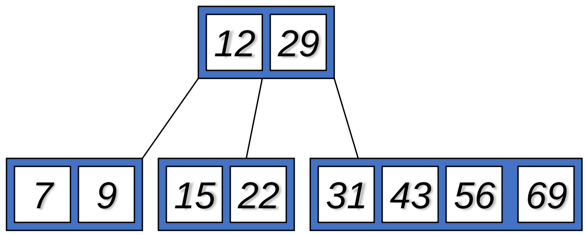


Not enough key. **Merge** nodes.

26

Example

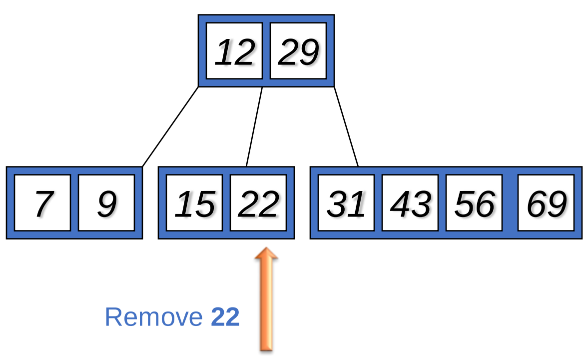
Given a B-tree order 5:



fit@hcmus

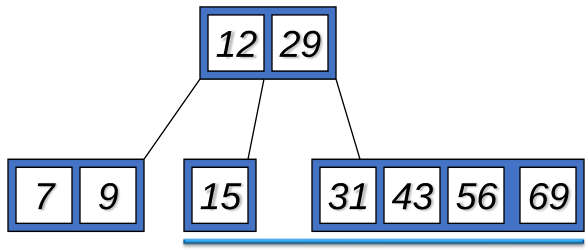
Example

Given a B-tree order 5:



Example

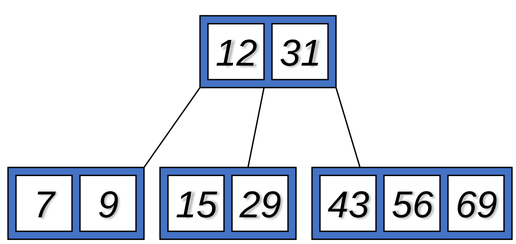
Given a B-tree order 5:



(**Redistribute**) Borrow key from neighboring nodes

Example

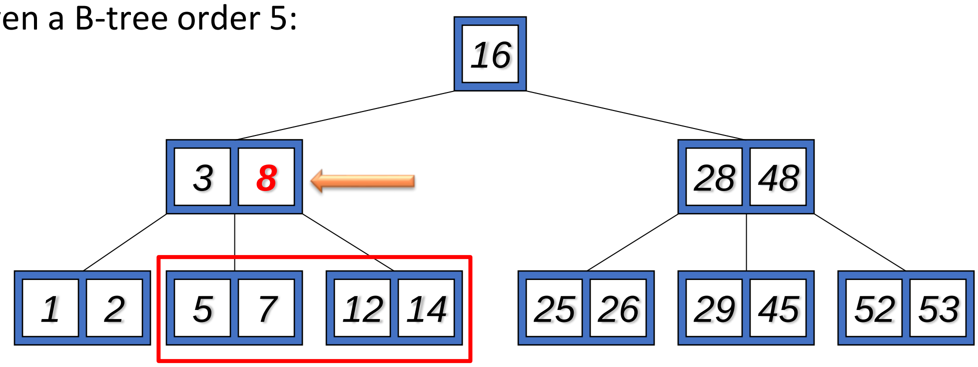
Given a B-tree order 5:



fit@hcmus

Example

Given a B-tree order 5:



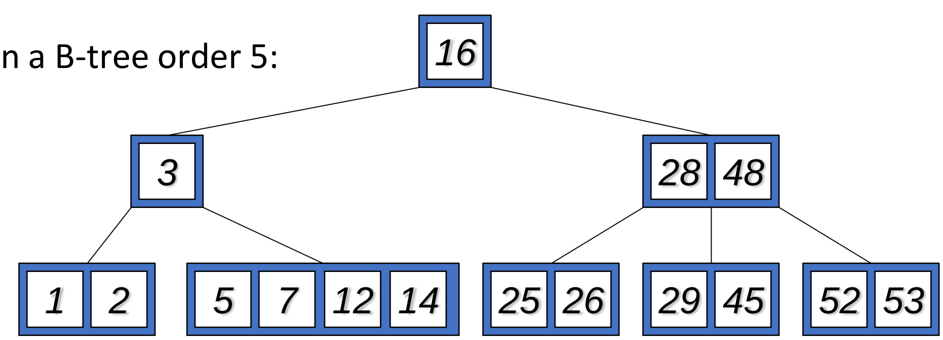
Remove 8 → Merge 2 child nodes of 8

31

fit@hcmus

Example

Given a B-tree order 5:

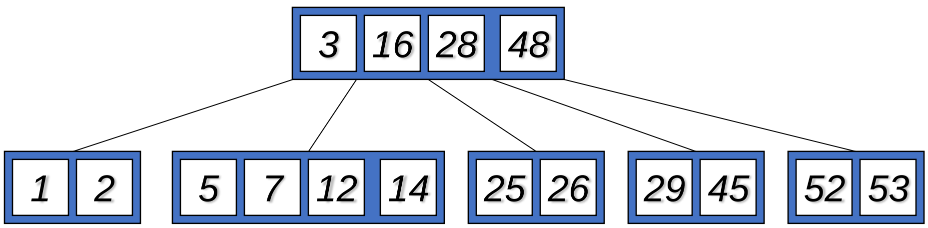


Not enough keys at node 3.

32

Example

Given a B-tree order 5



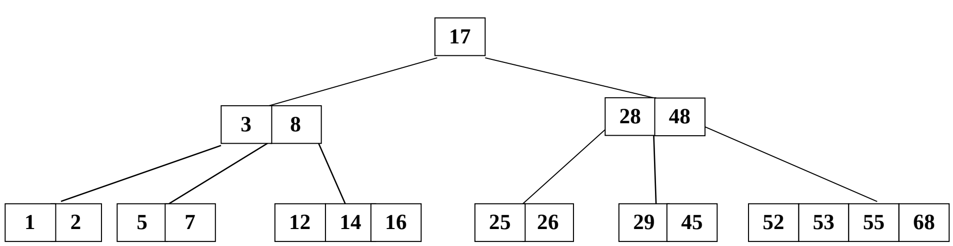
A new root node

Decrease height of the tree

Quiz

Given the following B-tree order 5

- Remove 28, then remove 48



B-Trees & Efficiency

fit@hcmus

- Used in Mac, NTFS, OS2 for file structure.
- Allow insertion and deletion into a tree structure, based on $\log_m n$ property, where *m* is the order of the tree.
- The idea is that you leave some key spaces open. An insertion of a new key is done using available space (most cases).
  - Less dynamic than our typical Binary Tree
  - Ideal for disk-based operations.

36

B-Trees & Efficiency

fit@hcmus

- In practical applications, B-Trees of large order (e.g., m = 128) are more common than low-order B-Trees such as 2-3 trees.

37

Questions and Answers