Data structures and Algorithms

B-TREES

Nguyễn Ngọc Thảo
nnthao@fit.hcmus.edu.vn

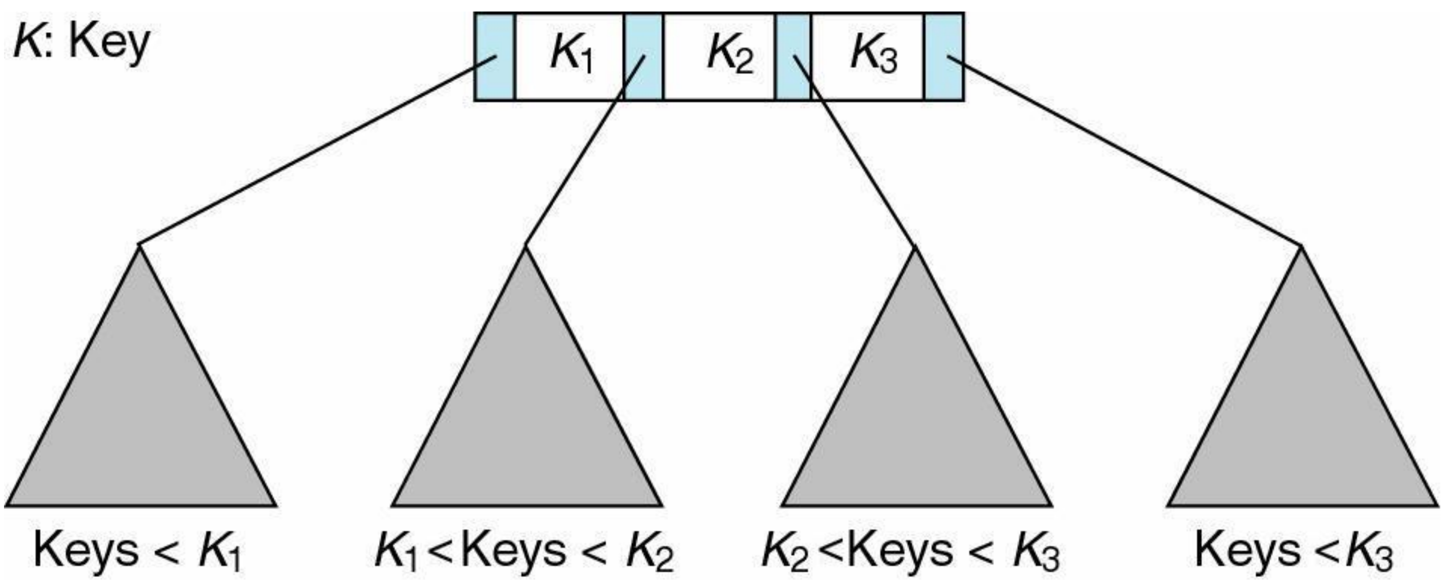
m-way Trees

The needs of m-way trees

- Binary trees allow for efficient manipulation in memory due to its simple representation.
  - Each node has one data item and at most two branches.
- However, it could not satisfy several practical needs.
  - Store data neatly in external storage $\rightarrow$ each unit to be stored should contain more than one data item.
  - Reduce cost for basic operations (i.e., search, insertion or removal) $\rightarrow$ a better tree architecture is required.
  - Optimal search for a data time $\rightarrow$ balanced search tree
- m-way trees are solution to the above issues.

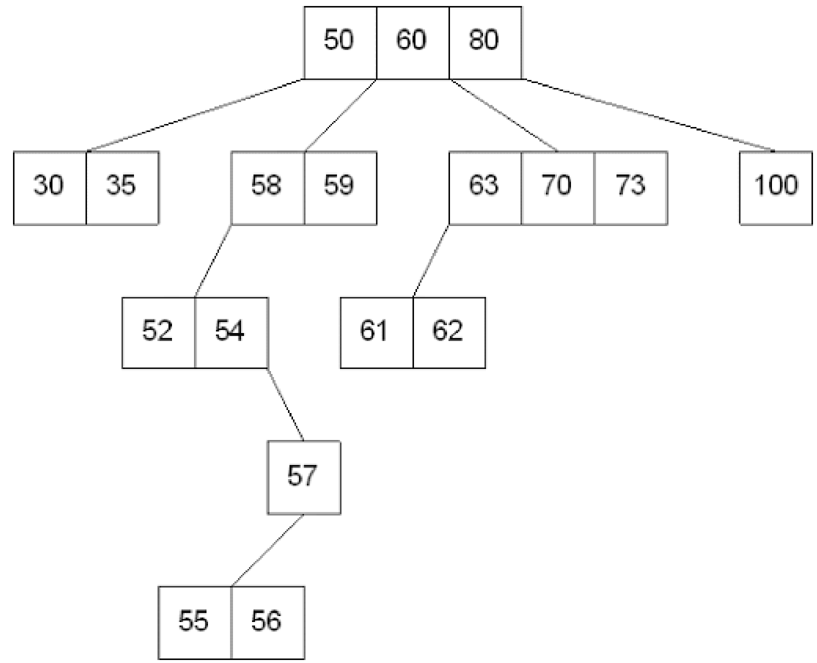
m-way trees: A definition

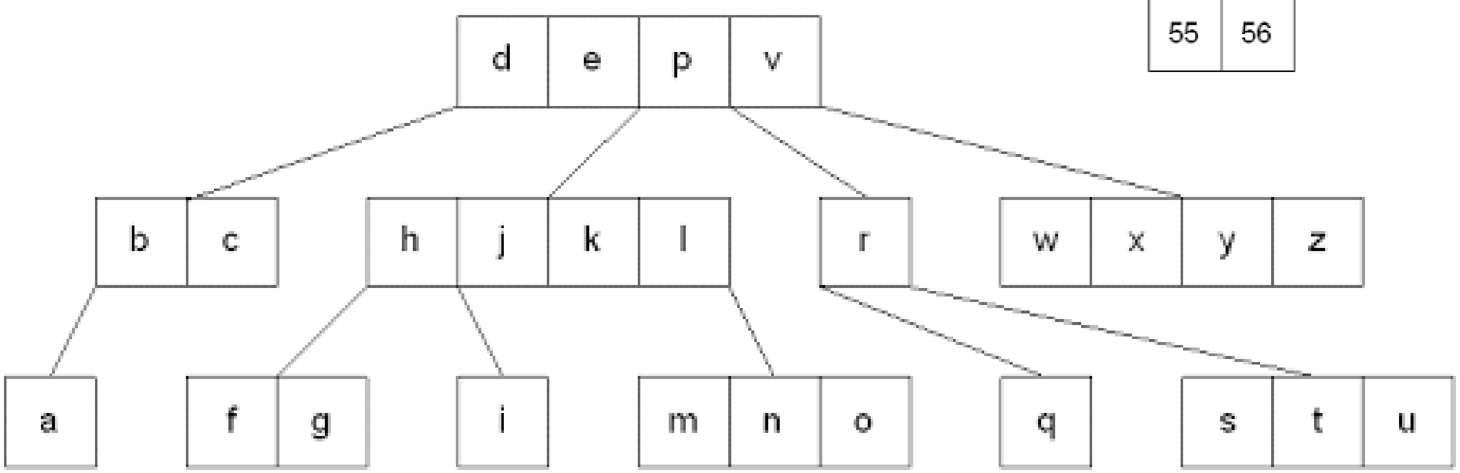
- A m-way tree has every internal node of at most m children and at most $(m-1)$ data items.
- The data items within a node are sorted in ascending order.
  - For any i^{th} data item, it is larger than every item in the i^{th} subtree, and smaller than every item in the $(i+1)^{th}$ subtree.



Example: An example of m-way tree

4-way tree





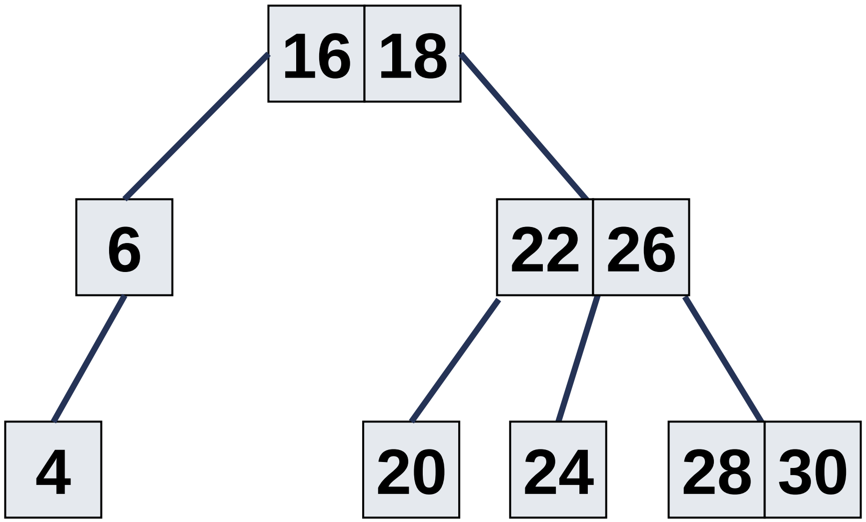
5-way tree

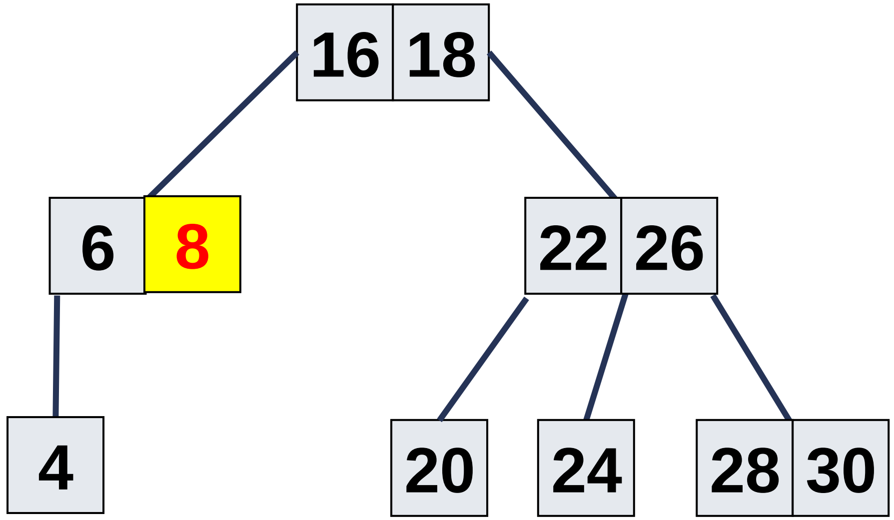
Insertion in a m-way tree

- Let $\boldsymbol{v}$ be the item to be inserted into a m-way tree.
- Traverse the tree until an empty subtree is found
- If the parent still has empty slot(s), insert $\boldsymbol{v}$.
- Otherwise, create a new node and insert $\boldsymbol{v}$ into that node.

Example: Insert data item to a 3-way tree

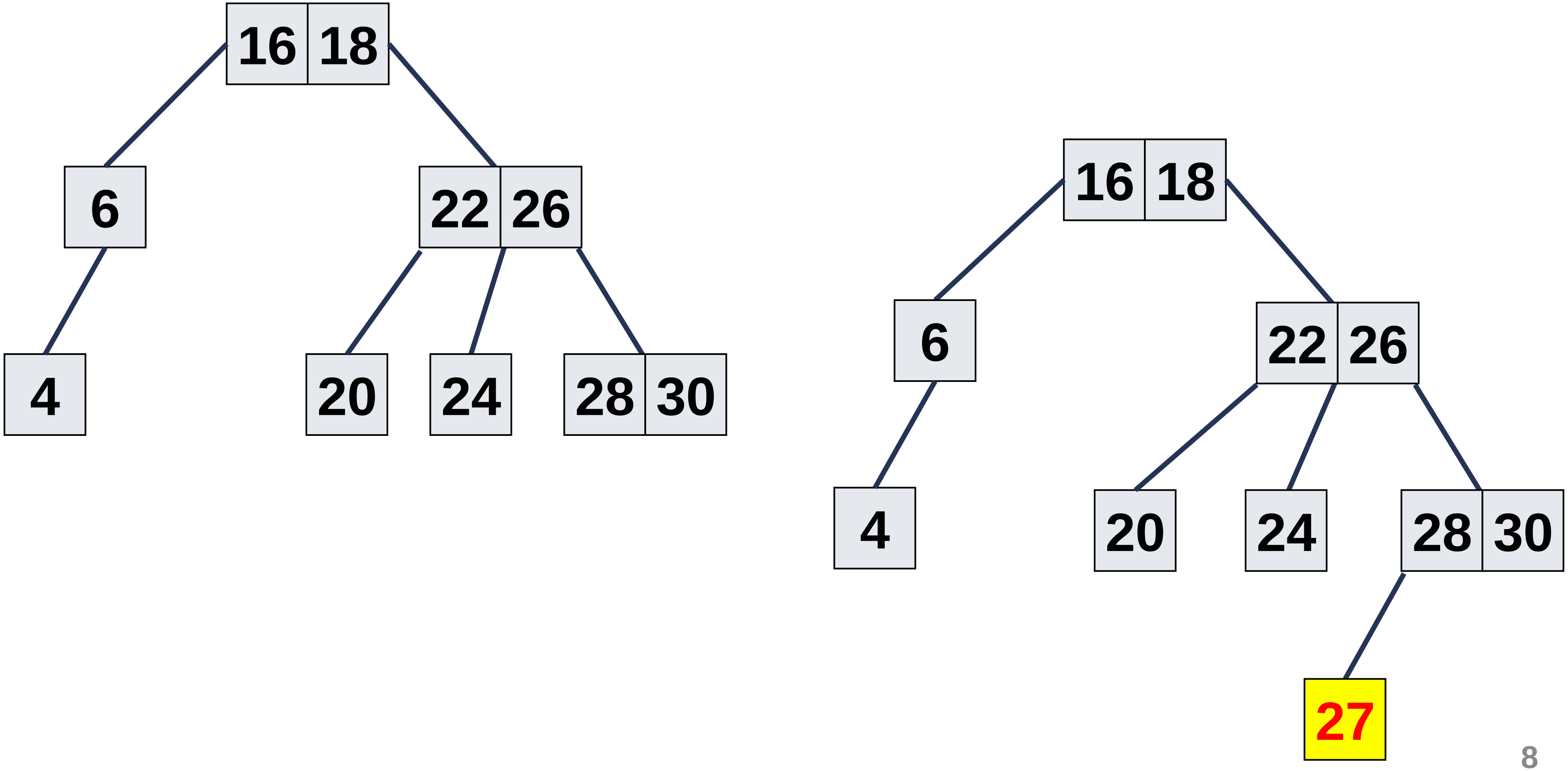
Insert data item 8: Still an empty slot at node ⟨6⟩.





Example: Insert data item to a 3-way tree

Insert data item 27: No empty slot at node ⟨28,30⟩.

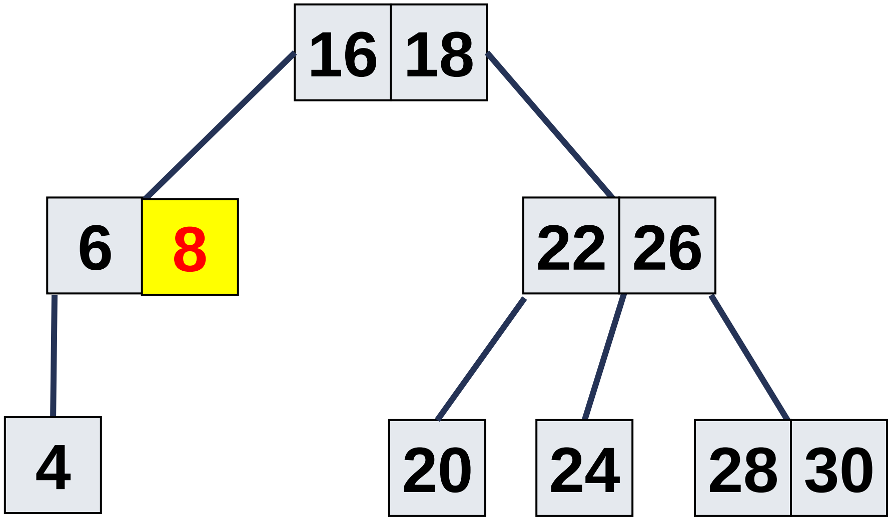


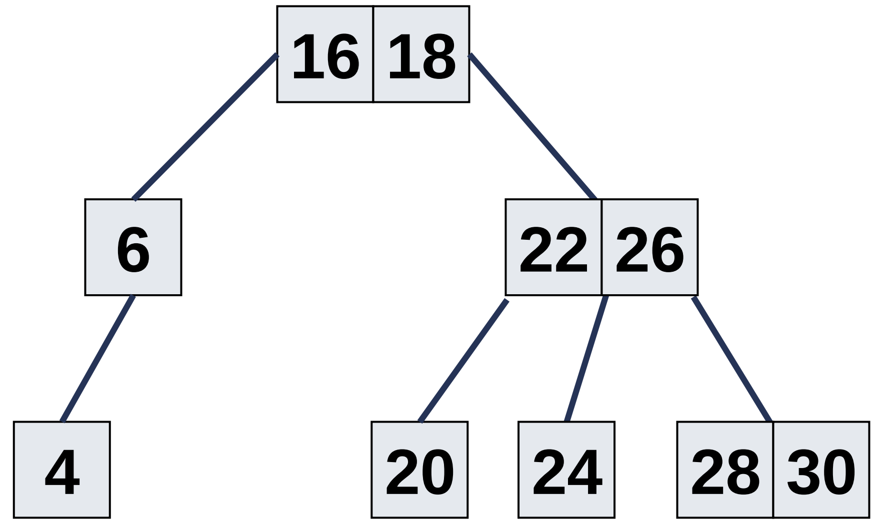
Removal in a m-way tree

- Let $\boldsymbol{v}$ be the item to be remove from a m-way tree.
- If $\boldsymbol{v}$ has no child (i.e., it is in between two empty subtrees) then delete $\boldsymbol{v}$.
- Otherwise, find a substitution for $\boldsymbol{v}$ which is either an in-order predecessor or an in-order successor.

Example: Remove data item from a 3-way tree

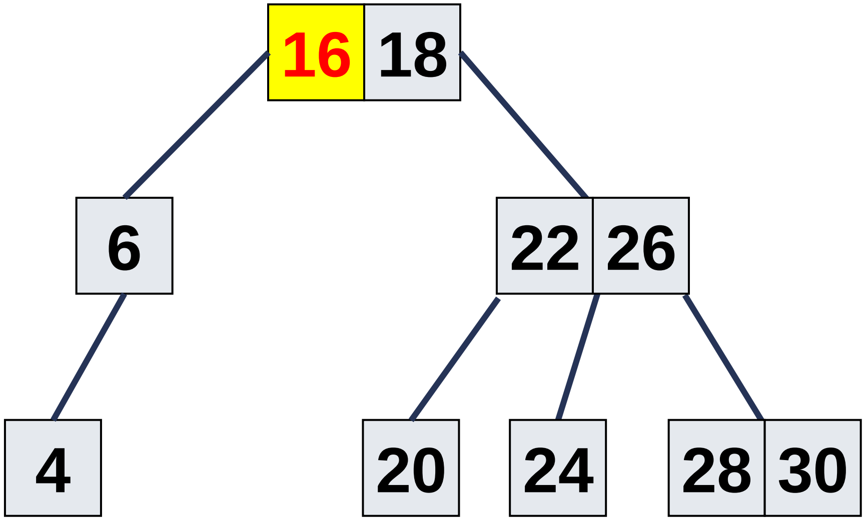
Remove data item 8: Node ⟨6,8⟩ does not have the 2^{nd} child and 3^{th} child.

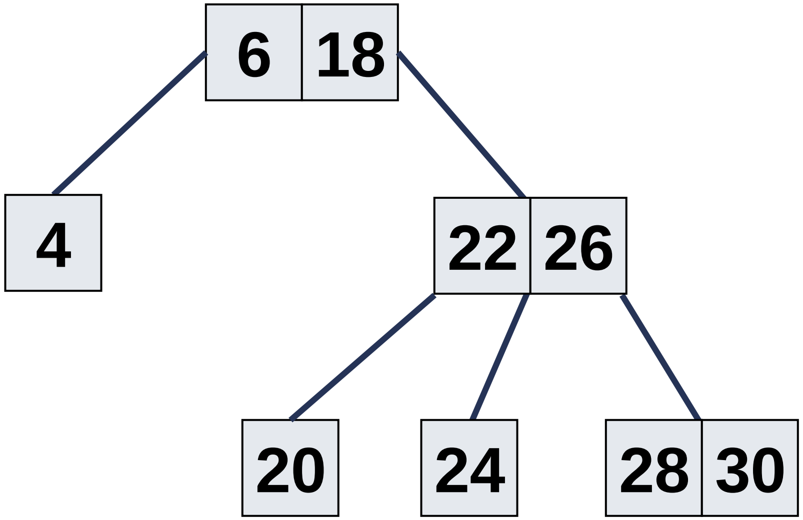




Example: Remove data item from a 3-way tree

Remove data item 16: The 1^{st} subtree is available.





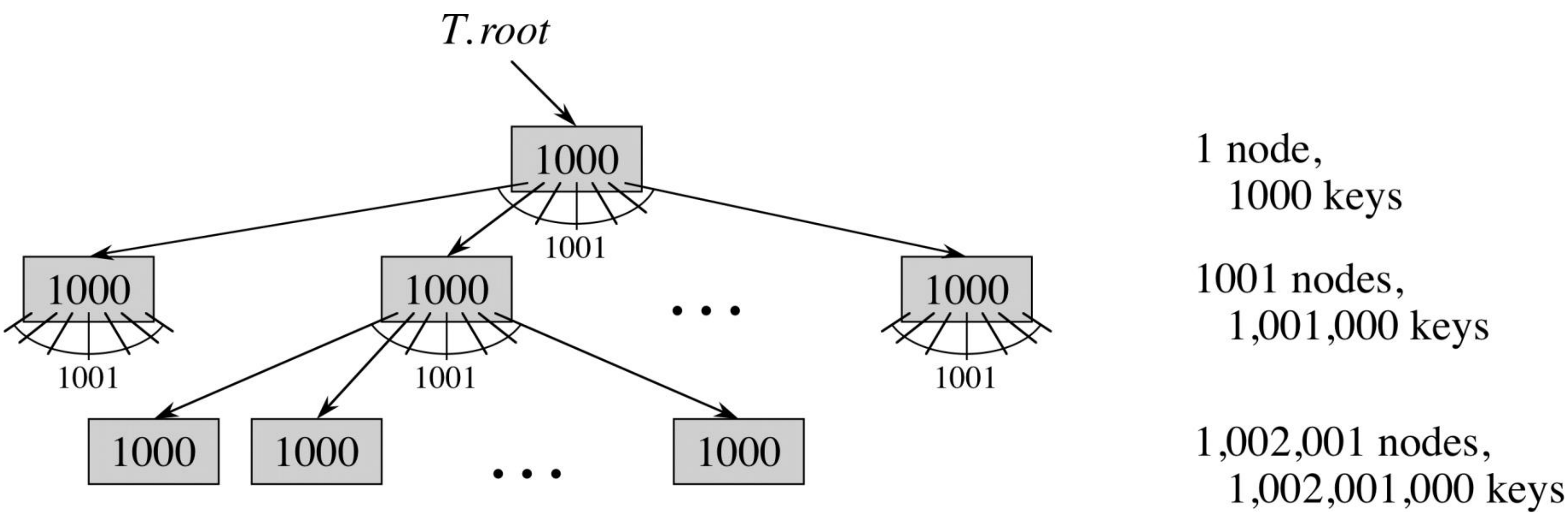
B-trees

B-trees (Bayer and McCreight, 1972)

- A B-tree is a self-balancing tree that maintains sorted data items and allows basic operations in logarithmic time.
  - Basic operations: search, sequential access, insertion, and deletion.
- It is a generalization of BST that allows for nodes with more than two children.
- B-tree is commonly used in databases and file systems.
  - It is well suited for storage systems that read and write relatively large blocks of data, such as disks.
  - Each node is filled at least 50%. In practice, it is normally ~70%.

Example: How many data items in a B-tree?

Data items in a B-tree of 1001 branches can be arranged in only 3 level $\rightarrow$ over one billions data items.

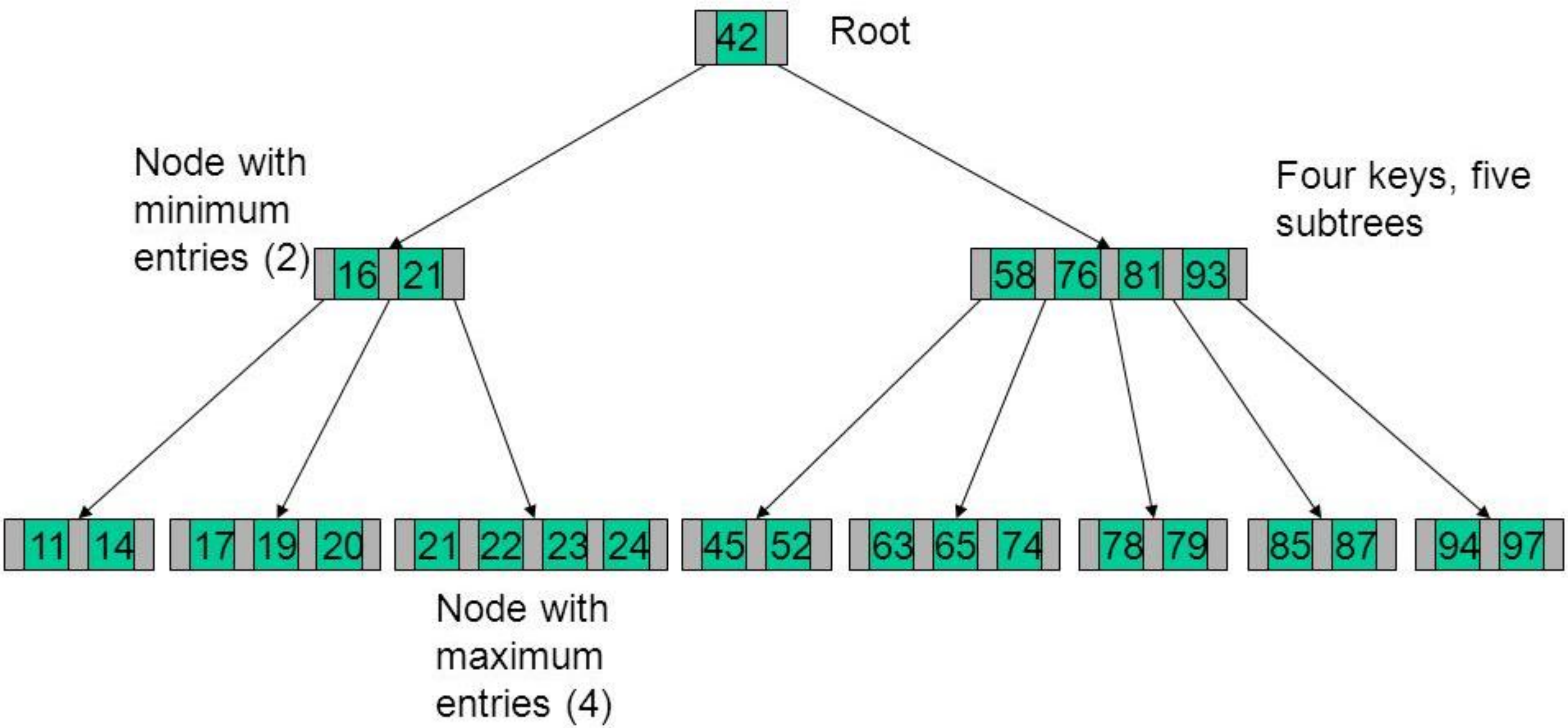


B-trees: A definition

- A B-tree of order m $(m > 2)$ is a m-way tree that satisfies the following conditions
  1. Every node has at most m children.
  2. Every non-leaf node (except root) has at least $\lceil m/2 \rceil$ 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 $k - 1$ items.
  5. All leaves appear in the same level.
- *Note that the above B-tree is derived from Knuth's definition, while that of Bayer and McCreight is slightly different.*

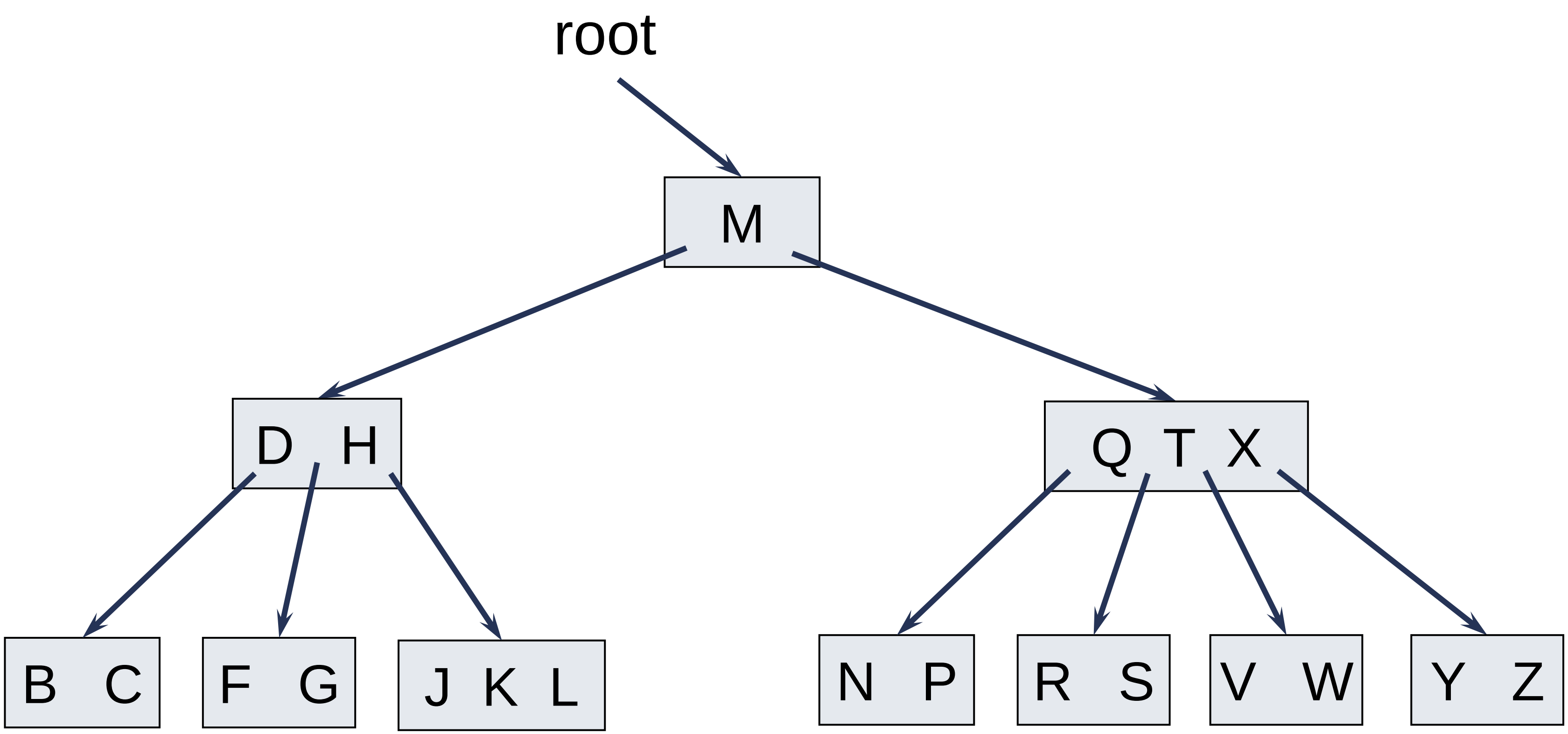
- 2-3 trees and 2-3-4 trees are derivations of B-trees when $m = 3$ and $m = 4$, respectively.

Example: B-tree of order 5

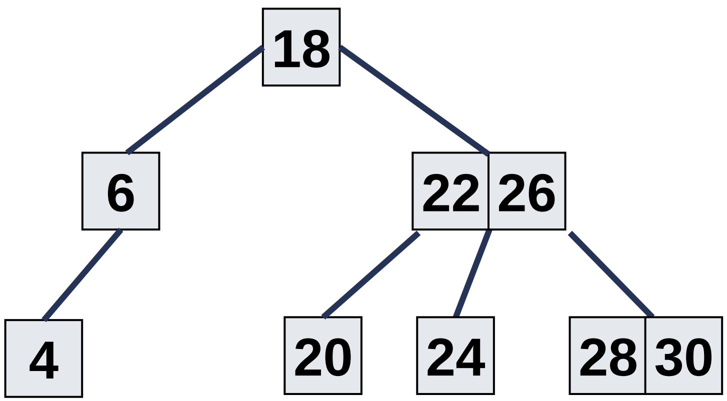


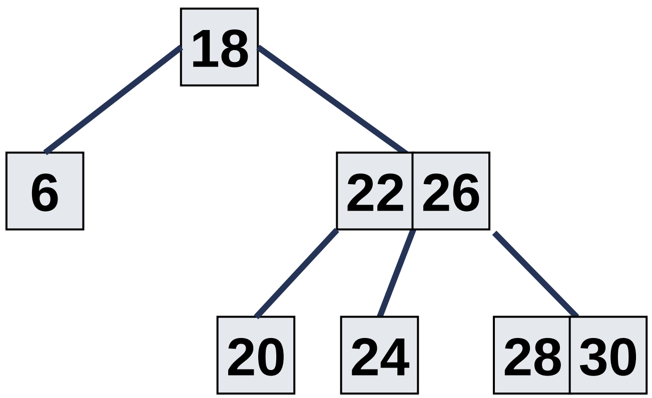
Min # of subtrees is 3 and max is 5.   Min # of data items is 2 and max is 4.

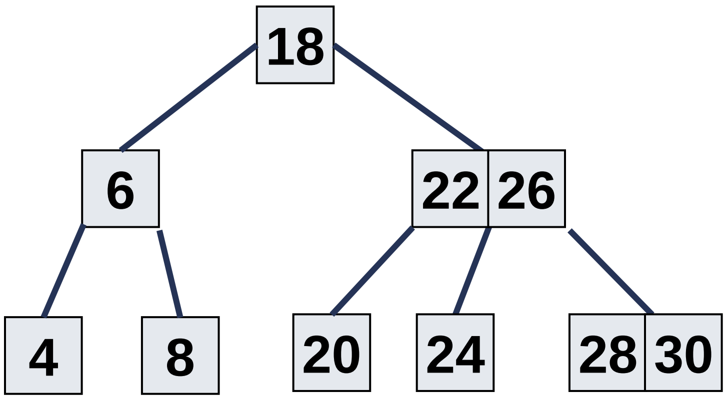
Example: B-tree of order 4

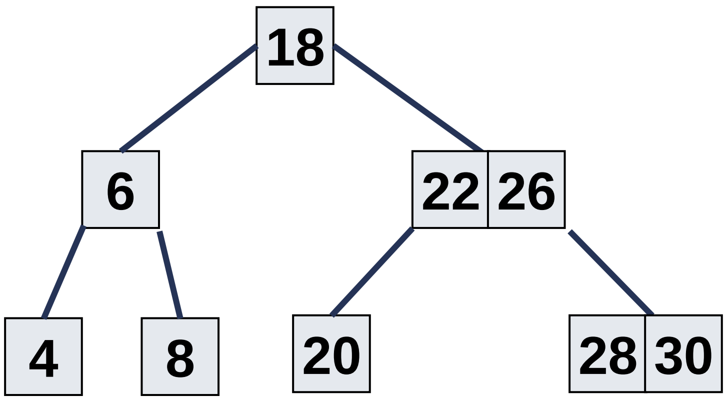


Checkpoint 01: Which of the following is a B-tree?









The height of a B-tree

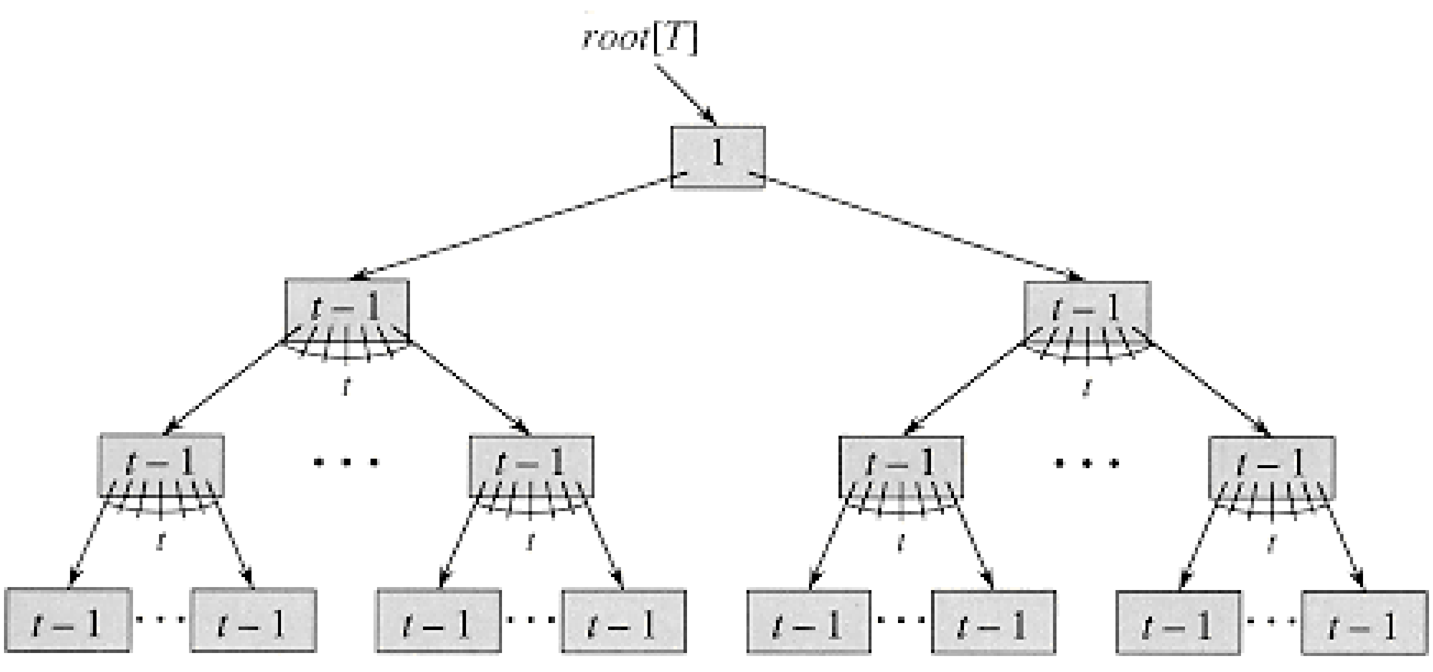
- Let $n \geq 0$ be the number of data items in the tree.
- Let m be the maximum number of children a node can have.
  - Each node can have at most $m - 1$ data items.
- The minimum height of a B-tree is $h_{min} = \lceil \log_m(n+1) \rceil$.

- Let d be the minimum number of children for an internal node. For an ordinary B-tree, $d = \lceil m/2 \rceil$
- The maximum height of a B-tree is $h_{max} = \left\lfloor \log_d \left(\frac{n+1}{2} \right) \right\rfloor + 1$.

* *Note that the height h starts from 1.*

The height of a B-tree

- In the worst case the root has only one key and two children.
- Every other node has $d-1$ items and d children.



Height	Number of items
1	1
2	2(d-1)
3	2d(d-1)
4	$2d^2$(d-1)

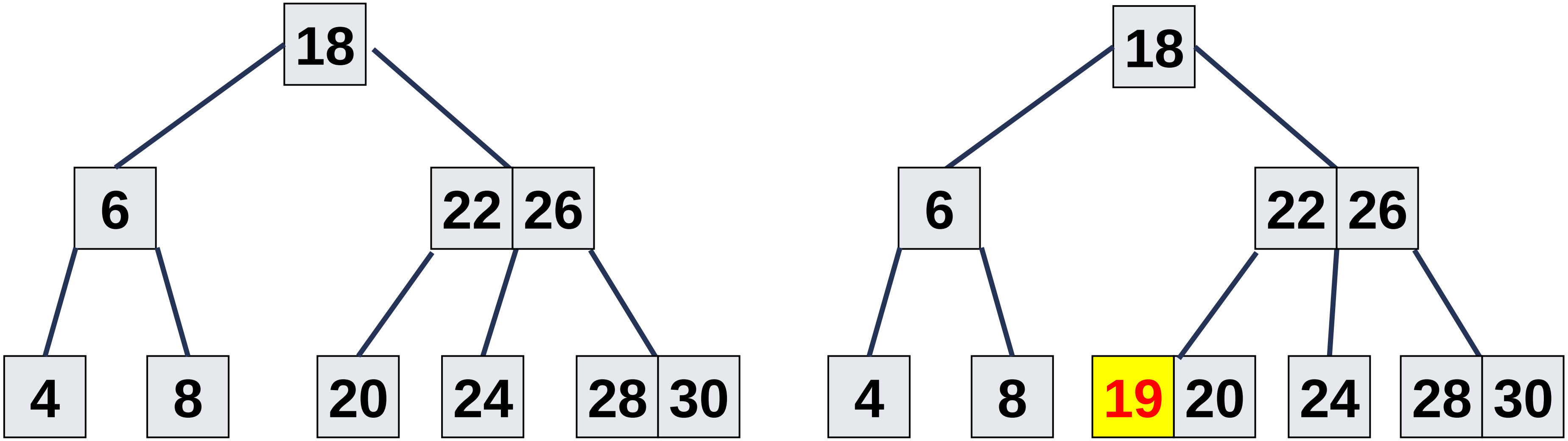
- The total number of items is $2d^{h-1}-1$.

Insertion in a B-tree

- Let $\boldsymbol{v}$ be the item to be inserted into a B-tree.
- Locate an appropriate leaf node to insert $\boldsymbol{v}$ by traversing the tree following the order of items.
- If the leaf node still has an empty slot, insert $\boldsymbol{v}$ to this node while maintaining the order of items.
- Otherwise, split the node.
- The split can be back-propagated to upper nodes.
  - Worst case: the root node is split and a new root is created.

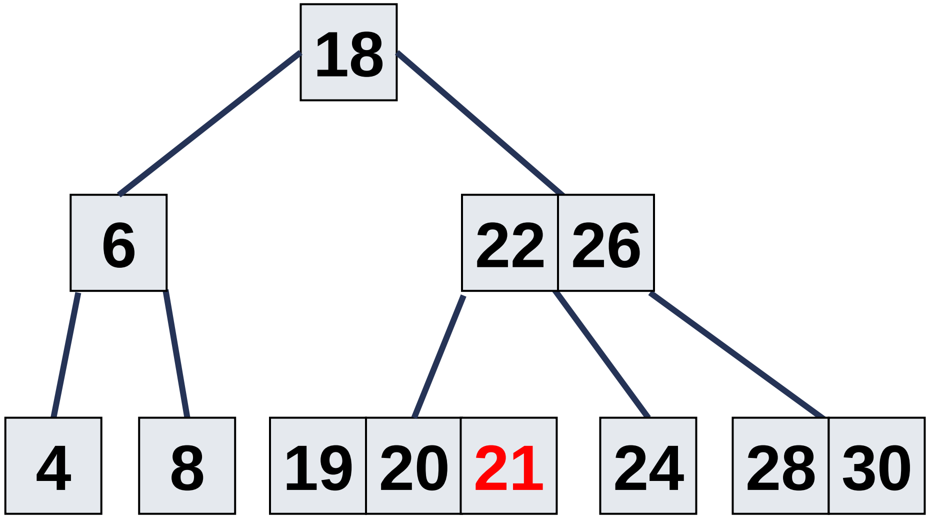
Example: Insert data item to a B-tree of order 3

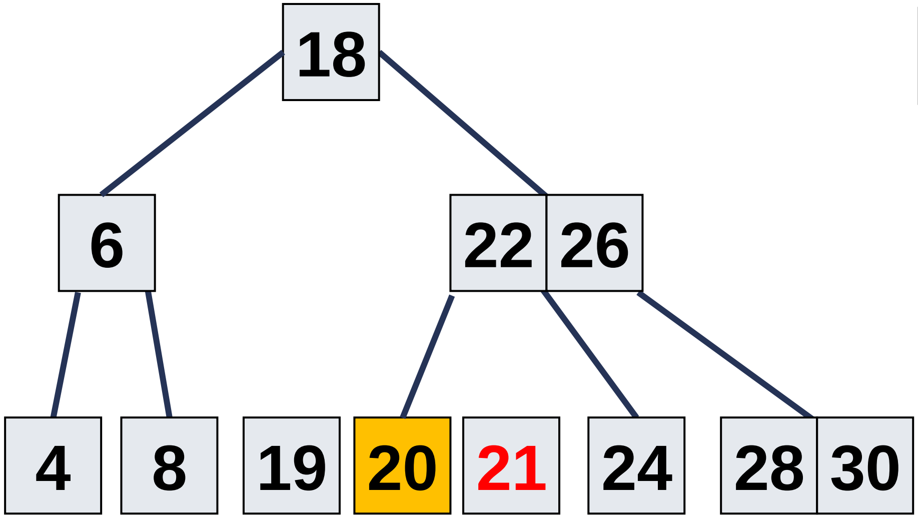
Insert data item 19: The leaf node ⟨20⟩ still has an empty slot.

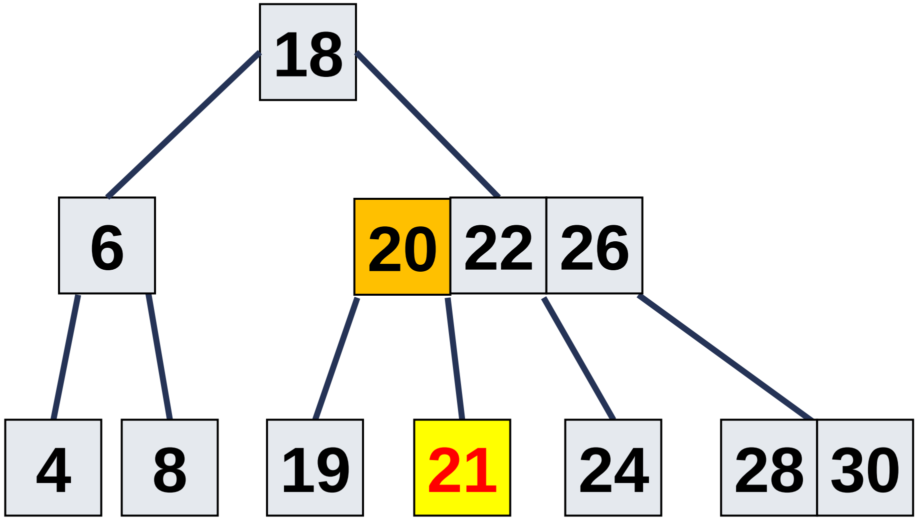


Example: Insert data item to a B-tree of order 3

Insert data item 21: The leaf node is full → split the node → move the middle item 20 to its parent node

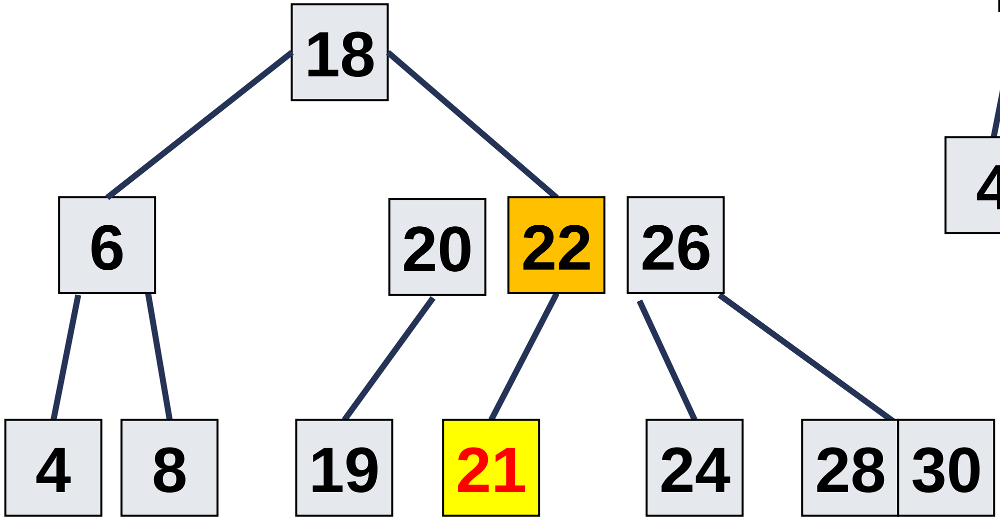


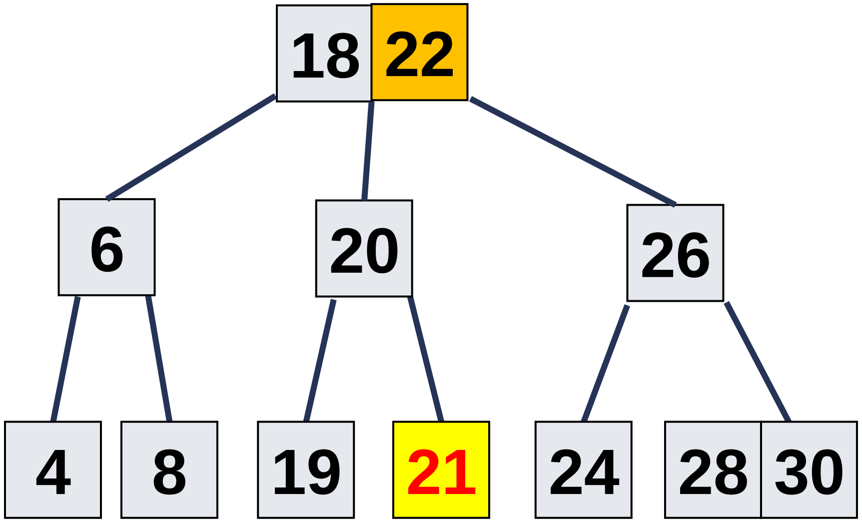




Example: Insert data item to a B-tree of order 3

Insert data item 21: ….The parent node becomes full too → split it using the same procedure.



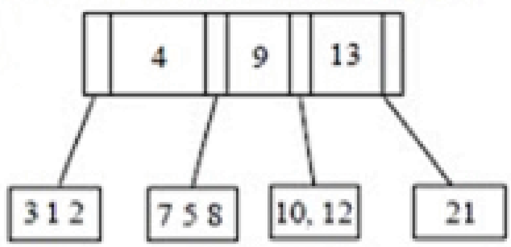


Checkpoint 02a: Insertion in a B-tree

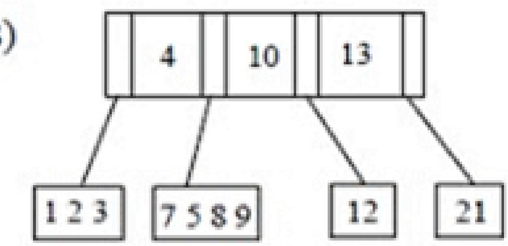
What is the resultant B-tree of order 4 after inserting the keys in the following order? Insert: 5, 3, 21, 9, 1, 13, 2, 7, 10, 12, 4, 8.

Knowing that the “middle key” is right-biased.

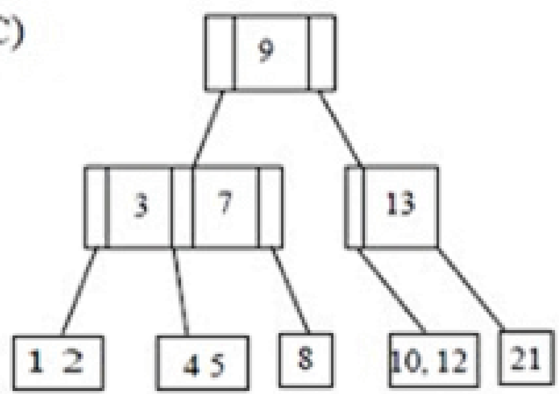
(A)



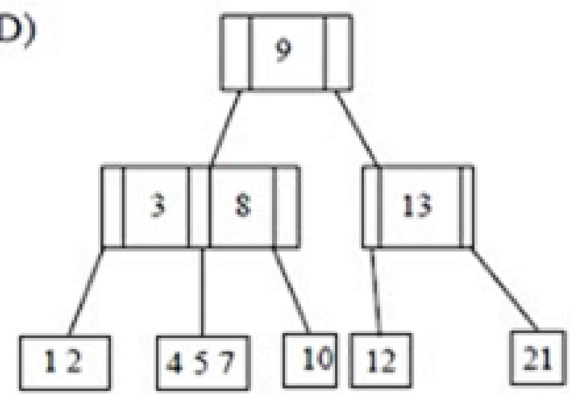
(B)



(C)



(D)



Checkpoint 02b: Insertion in a B-tree

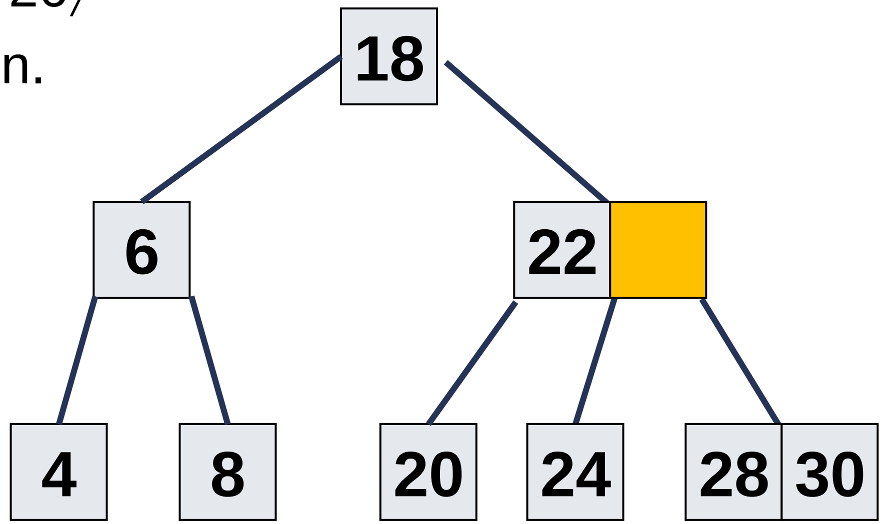
Insert the keys 78, 52, 81, 40, 33, 90, 85, 30 and 38 in this order in an initially empty B-tree of order 3.

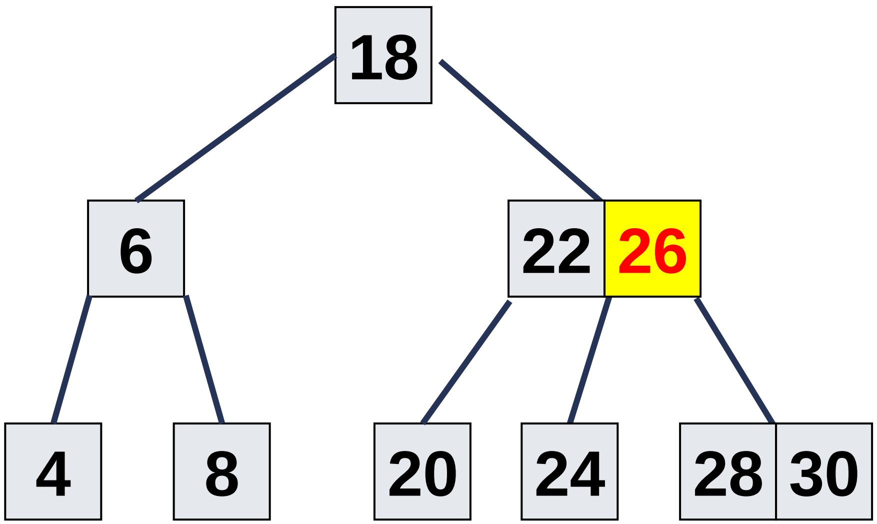
Removal in a B-tree

- Let $\boldsymbol{v}$ be the item to be removed from a B-tree.
- If $\boldsymbol{v}$ is at the leaf node
  - If the number of items remained after removal $\geq \left\lceil \frac{m}{2} \right\rceil - 1 \rightarrow$ **Stop!**
  - Otherwise, borrow an item from a sibling node that has more than $\left\lceil \frac{m}{2} \right\rceil - 1$ items.
  - If there is no such sibling node, merge the node with its sibling node and take one data item down from the parent node.
  - If the parent node becomes insufficient, apply the same procedure.
- If v is NOT at the leaf node
  - Find a substation for $\boldsymbol{v}$ at some leaf node and delete the actual slot at that leaf node.

Example: Remove data item from a B-tree of order 3

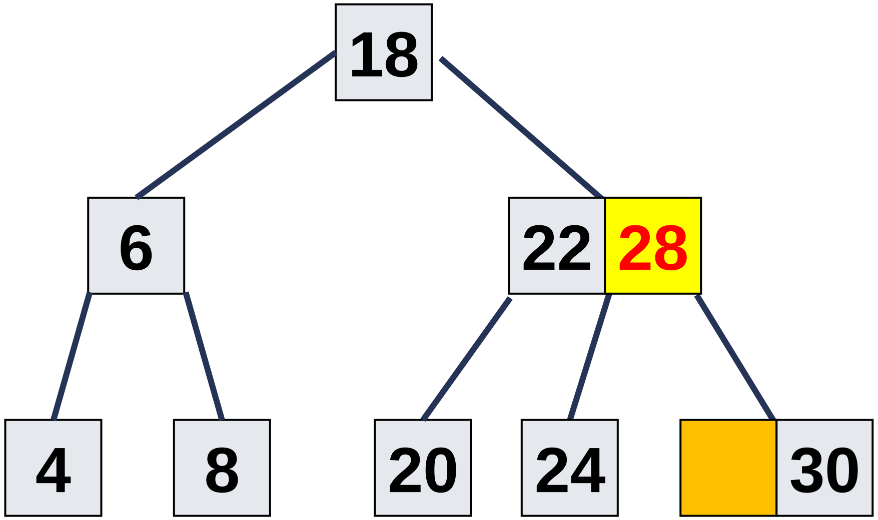
Remove data item 26: The node ⟨22, 26⟩ is not a leaf node → find a substitution.

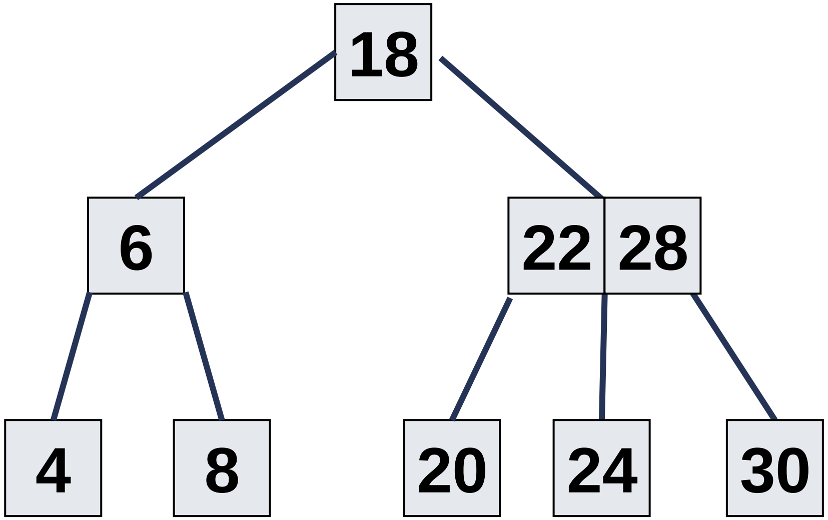




Example: Remove data item from a B-tree of order 3

Remove data item 26: Replace 26 by 28 $\rightarrow$ delete the slot that previously contains 28.



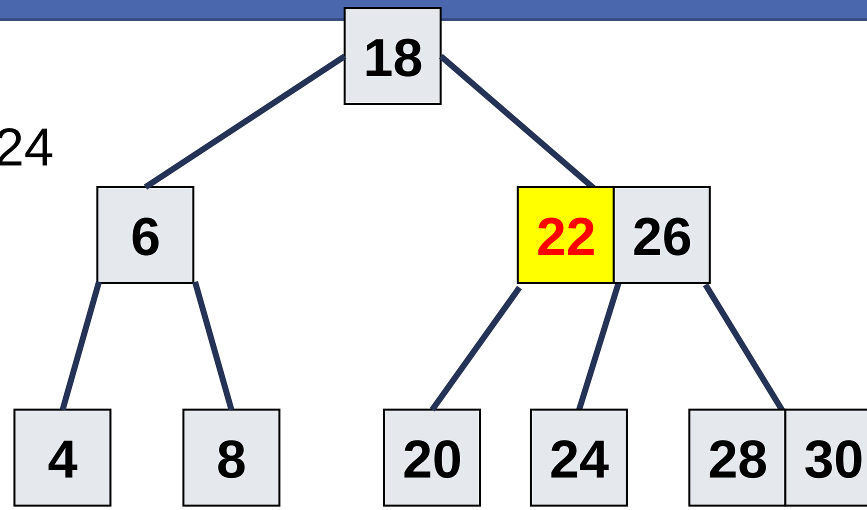


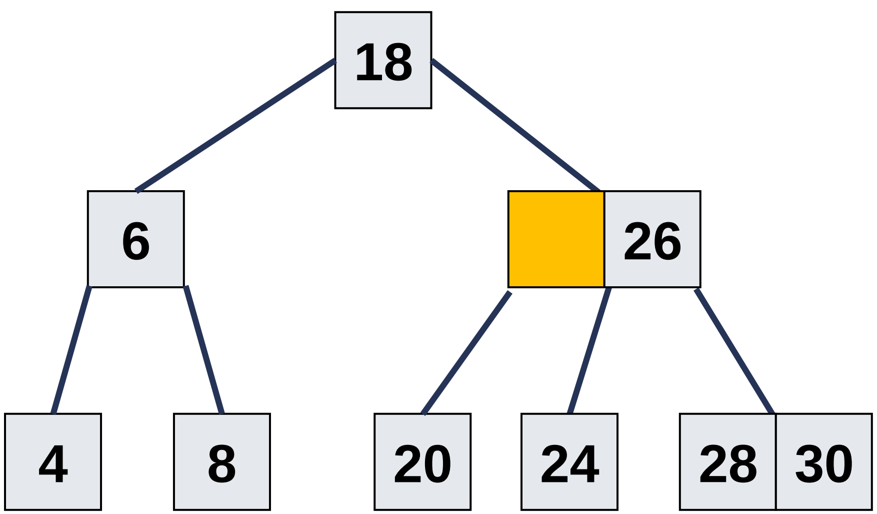
Example: Remove data item from a B-tree of order 3

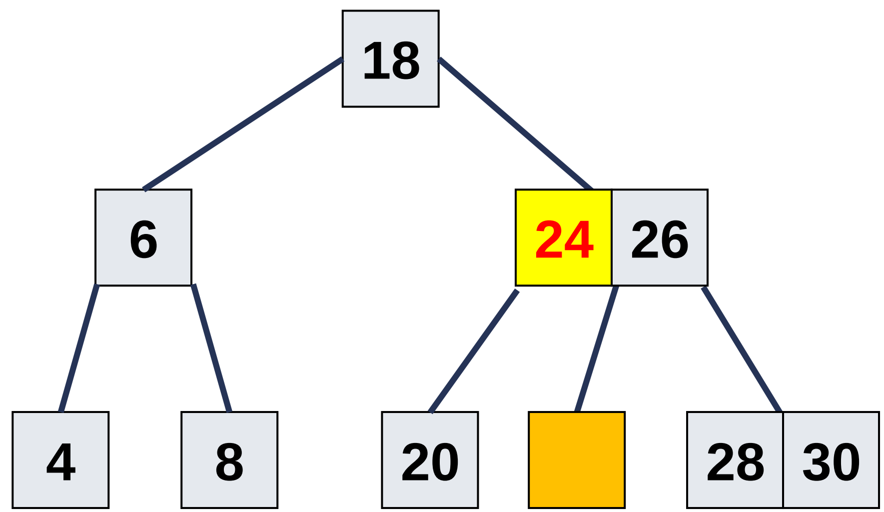
Remove data item 22: Replace 22 by 24

→ the leaf node becomes insufficient.

.

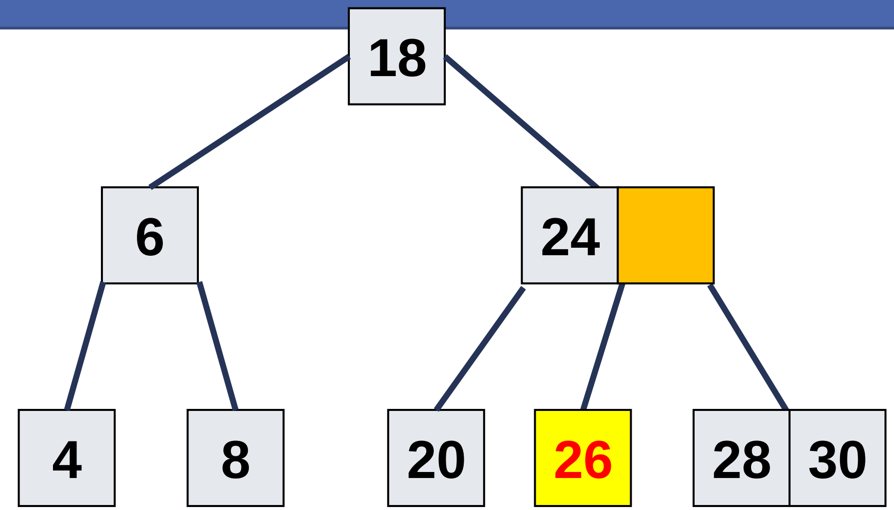




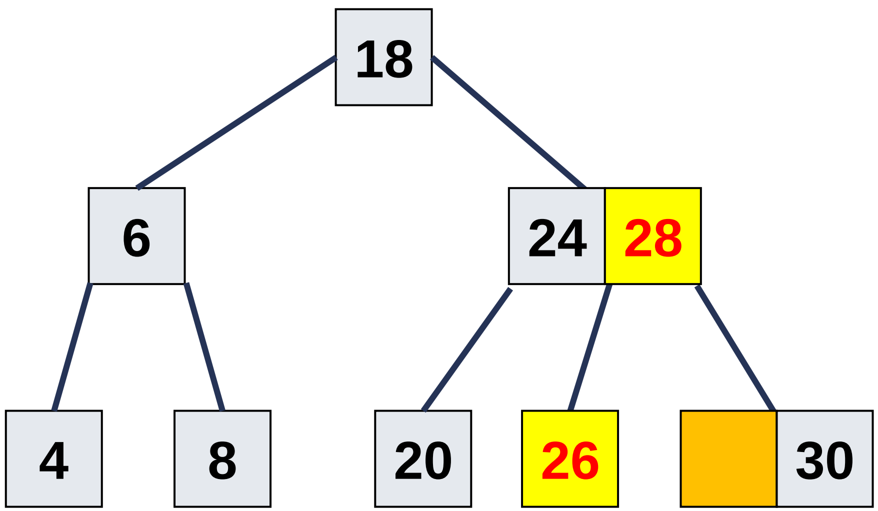


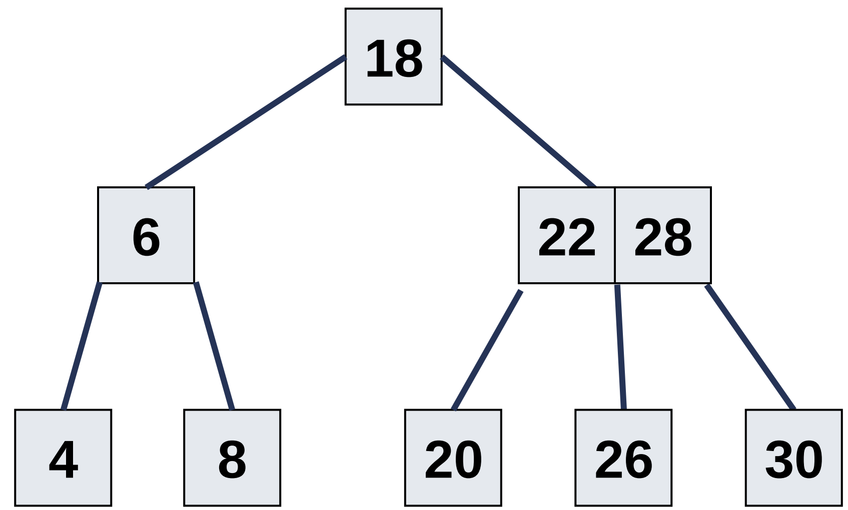
Example: Remove data item from a B-tree of order 3

Remove data item 22: ……borrow 28 from a sibling node



.

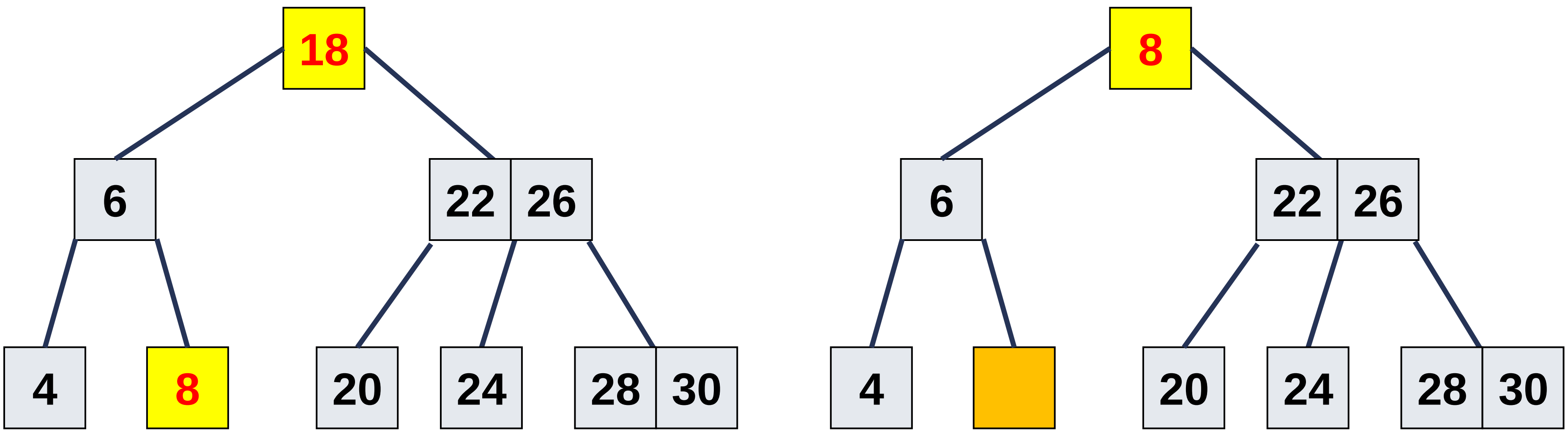




Example: Remove data item from a B-tree of order 3

Remove data item 18: Replace 18 by 8…the leaf node become insufficient.

.

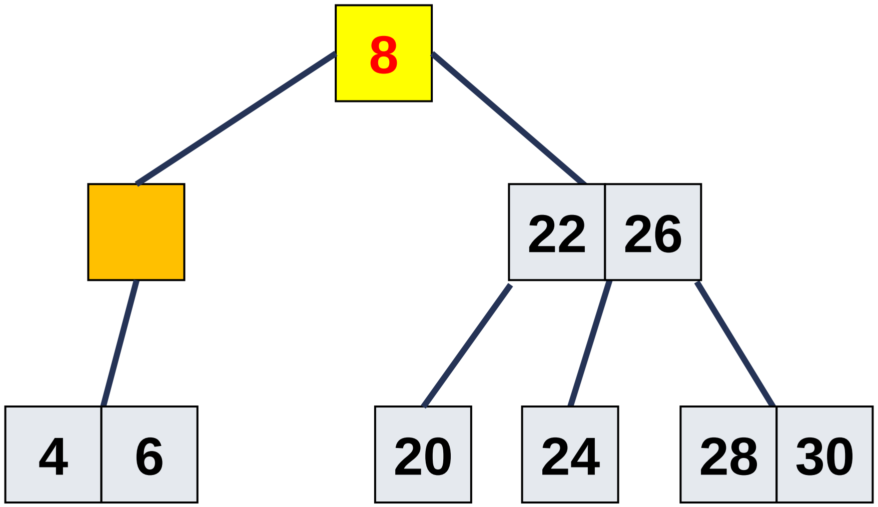


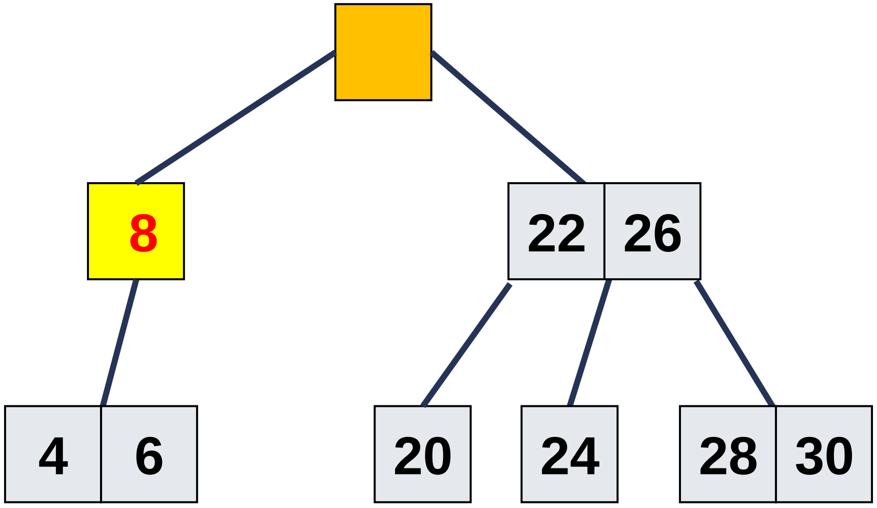
Example: Remove data item from a B-tree of order 3

Remove data item 18: …merge the two sibling and the data items in the parent node → the parent node becomes insufficient.

.

.





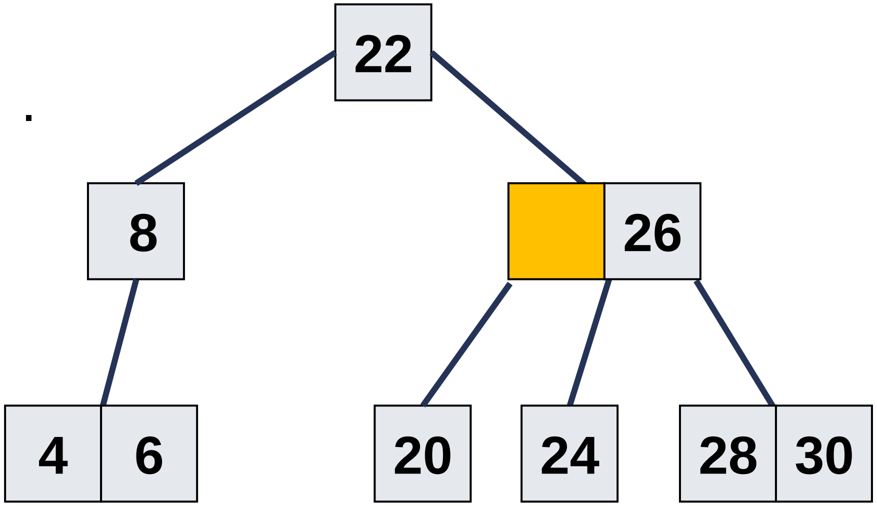
Example: Remove data item from a B-tree of order 3

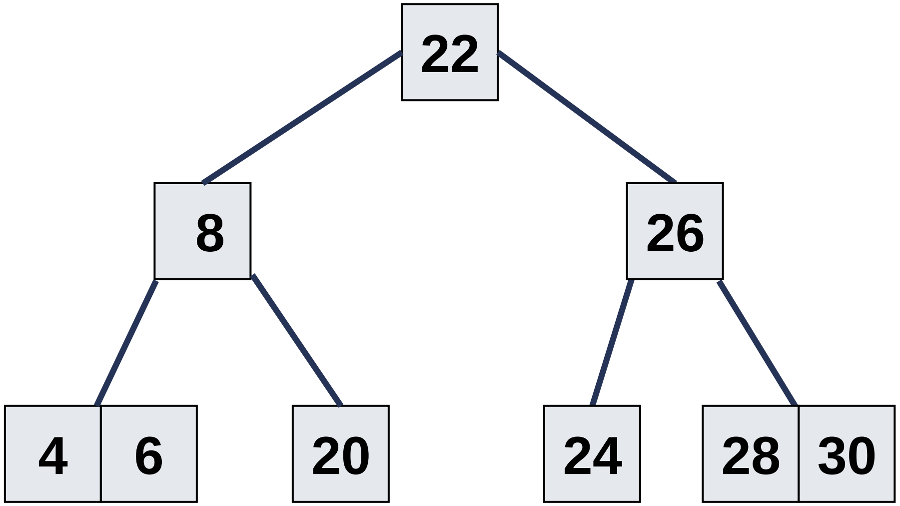
Remove data item 18: …borrow 22 in the sibling node and move corresponding subtrees.

.

.

.





B-trees: Implementation

```
class BNode{
    bool leaf;                  // A flag for the leaf node
    unsigned int nItems;        // The number of available data items
    ItemType keys[MAX-1];       // An array of data items
    // An array of pointers to subtrees
    unsigned long pointers[MAX];
    // Redundant data to completely fill the block
    char unused[K];
}
* MAX and K are predefined constants.
```

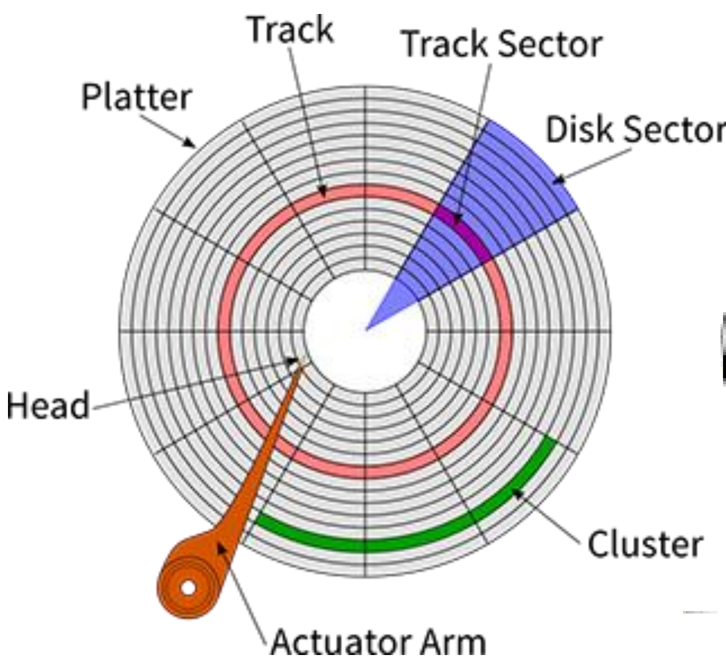
B-tree vs. m-way tree

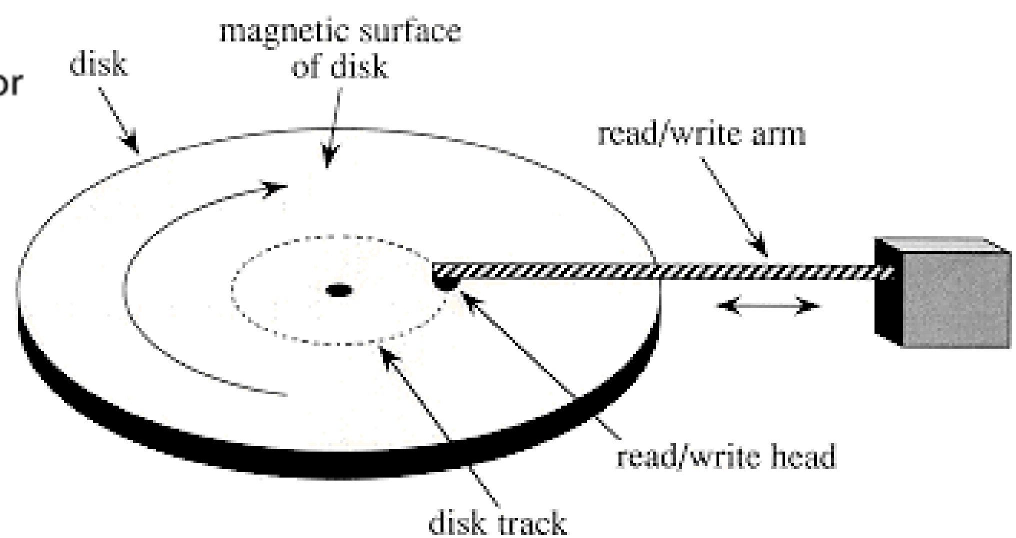
- m-way tree is not a balanced search tree.
  - Data insertion and removal are quite simple.
  - The tree grows toward the leaves.
- B-Tree is basically a balanced m-way tree.
  - Data insertion or removal may involve the split/merge of nodes.
  - It minimizes the number of accesses to external storage.
  - The tree grows towards the root.

Store a tree in an external storage

- It is essential to have an efficient search method on the external storage.
- Let t be the time to read/write a block
- t = the time to move the reader to corresponding block

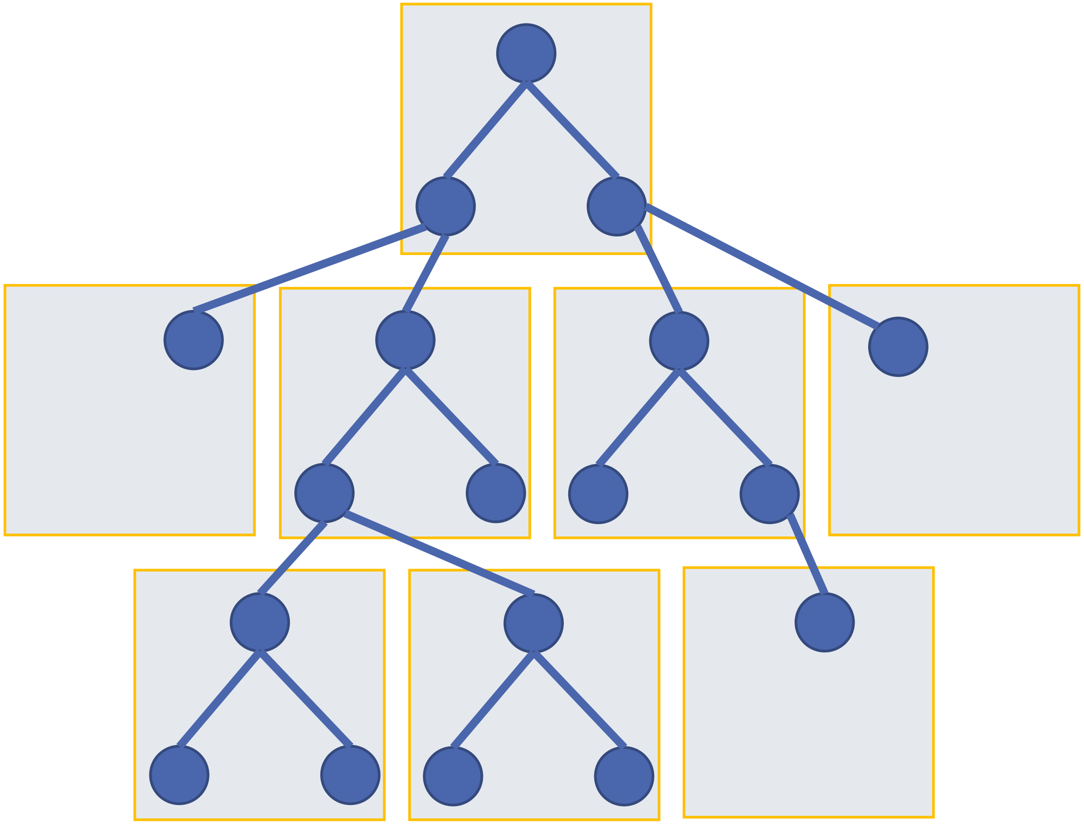
  + the time to read/write the block into memory





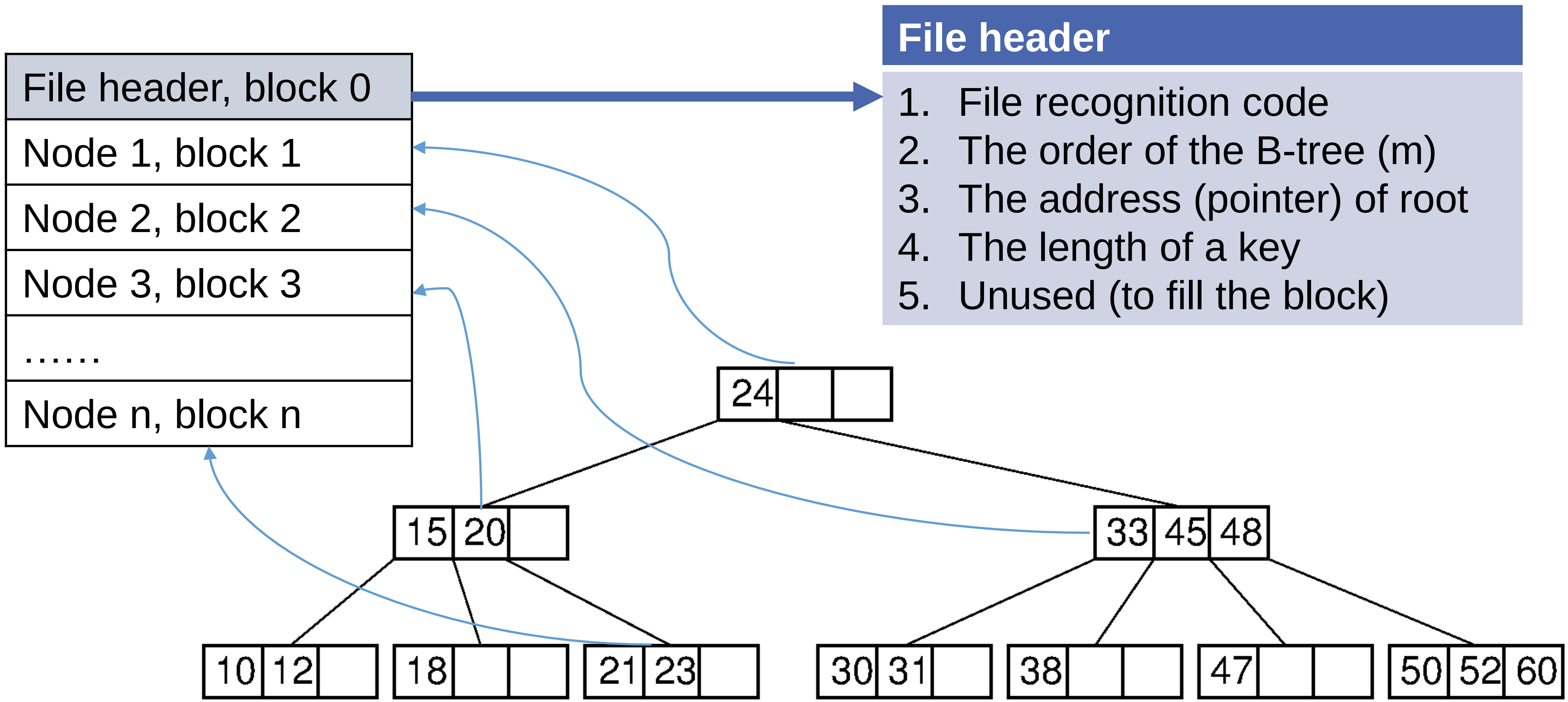
Store a tree in an external storage

- Here is a example of storing a binary tree, in which data items are grouped into blocks, to a disk storage.



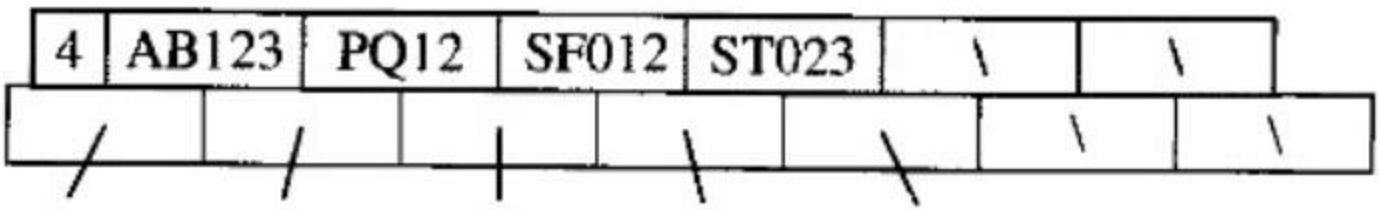
Store a tree in an external storage

- For a B-tree, the root should be cached for frequent access.
  - It is not necessary to perform the READ_ROOT
  - The WRITE_ROOT is only required when the root is modified.

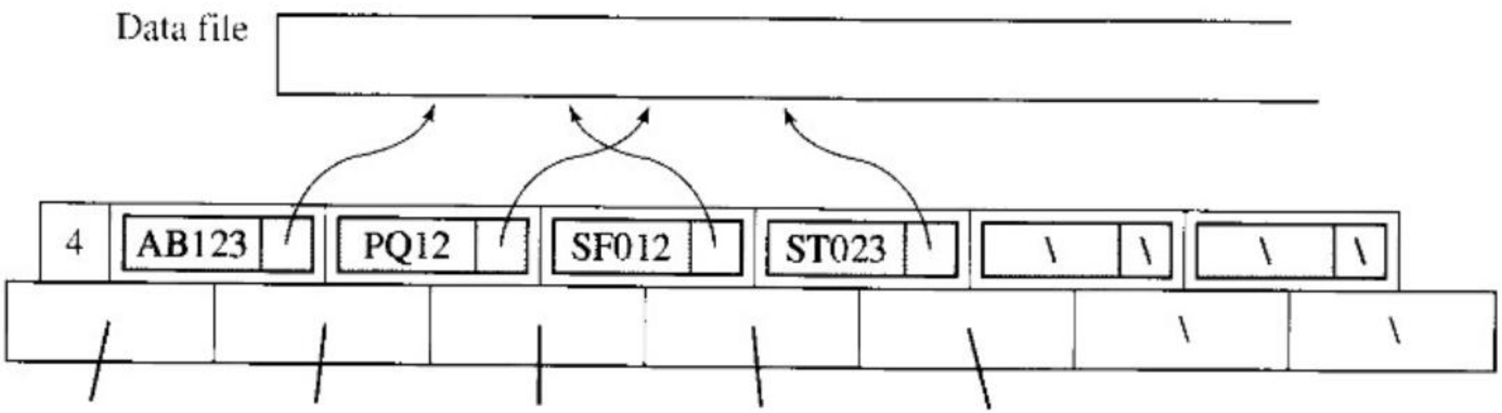


Store a tree in an external storage

- A B-tree with data items that have no supplementary info



- A B-tree with data items that have certain supplementary info



B+ Trees: A variant of B-tree

- Records (data) can only be stored in the leaf nodes while internal nodes can only store the key values.
  - Meanwhile, keys and records in a B-tree both can be stored in the internal and leaf nodes.
- The leaf nodes are linked together in the form of a singly linked lists to make the search queries more efficient.
- The internal nodes (keys to access records) are stored in the main memory, and leaves are in the secondary memory.

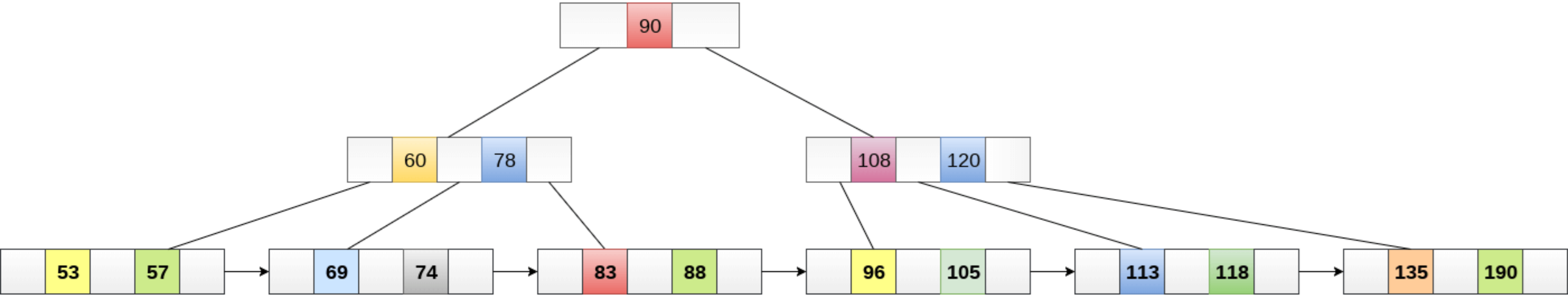
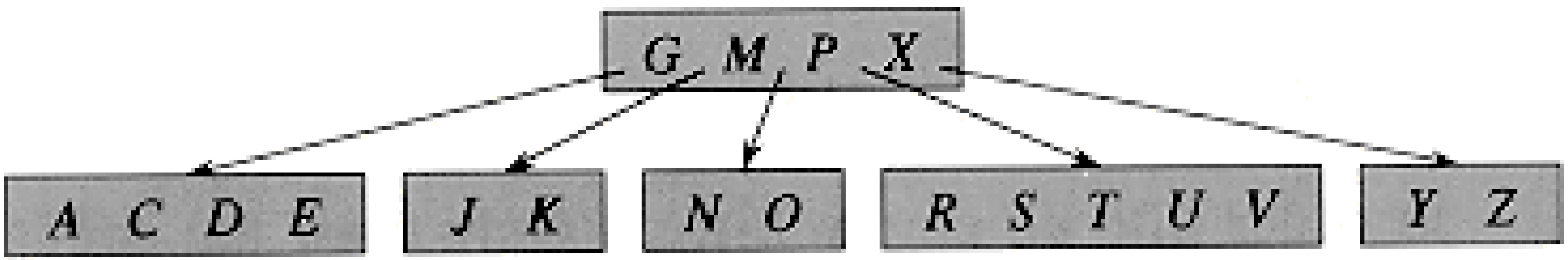


Image credit: Javatpoint

Exercises

01. Insertion in a B-tree

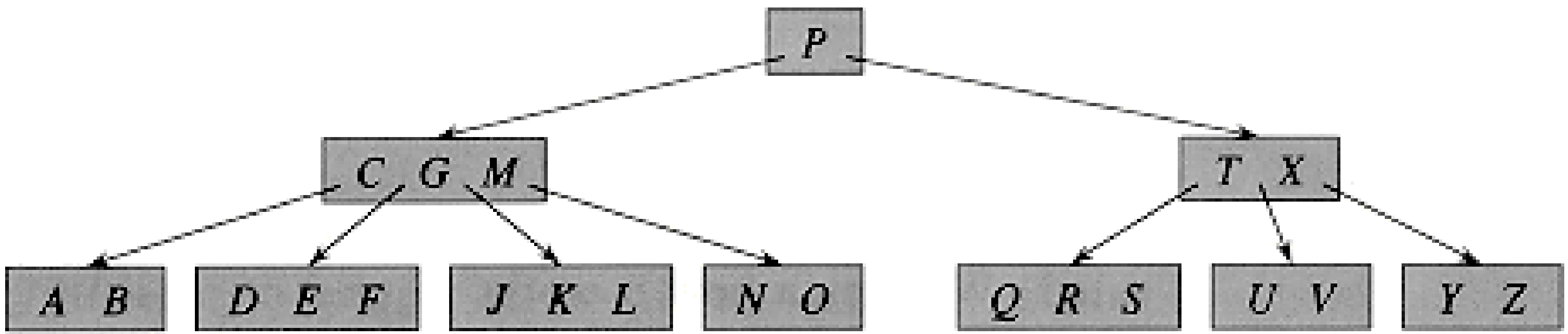
- Consider the following **B-tree of order 6**.



- Draw the resulting tree after each insertion of the following keys: B, Q, L, and F

02. Removal in a B-tree

- Consider the following **B-tree of order 6**.



- Draw the resulting tree after each deletion of the following keys: F, M, G, D, and B


...the end.