Non-Linear Data Structures

Multi-way Search Trees

Introduction:

- Binary search trees are good and efficient data structures for searching, when the number of nodes can fit within the computer's main memory.
- A binary search tree with a large number of keys, can use secondary storage devices to store nodes (each node holds one key), but access time will become a problem and needs to be minimized as follows:
  1. At each disk access, get a whole block of data (one big node).
  2. Use balanced multi-way search trees, to minimize tree height.

Example:

For six million keys:

- A binary tree will have a depth of $\log_2 10^6 \cong 20$.
- An m-way tree of m=100 will have a depth of $\log_{100} 10^6 = 3$.

Multi-way Search Tree Definition:

- A Multi-way search tree of order m, called m-way tree, is a tree in which all nodes are of degree <= m.

- A non-empty m-way tree has the following properties:
  1) A node, T, of the tree has the following structure:

P_0	K_1	P_1	K_2	P_2	…	K_{m-1}	P_{m-1}

  Where:
  P_i , $0 <= i <= (m-1)$ are pointers to subtrees of T and
  K_i , $1 <= i <= (m-1)$ are key values.

Multi-way Search Tree Properties:

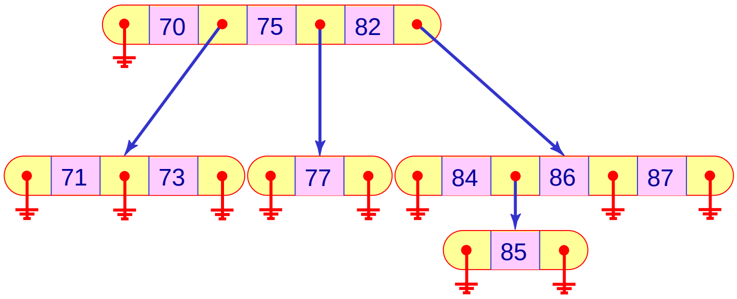
2) $K_i < K_{i+1}$, $1 <= i < (m-1)$.
3) All key values in subtree P_i are less than the key value K_{i+1}, $0 <= i < (m-1)$.
4) All key values in subtree P_{m-1} are greater than K_{m-1}.
5) The subtrees P_i, $0<= i <= m-1$ are also multi-way search trees of order m.
6) One or more of the subtrees of a node may be empty.

Example: A node of an m-way search tree with m=7:

P_0	K_1	P_1	K_2	P_2	K_3	P_3	K_4	P_4	K_5	P_5	K_6	P_6
•	10	•	25	•	40	•	45	•	60	•	100	•
<10		>10 <25		>25 <40		>40 <45		>45 <60		>60 <100		>100

Multi-way Search Tree Example:

An m-way tree
with m = 4:



Empty Slide

Non-Linear Data Structures

B-Tree
(Balanced Multi-way Tree)

B-tree (Balanced M-way Tree) Definition:

- A B-tree of order n, is an m-way (m = 2*n + 1) search tree that is either empty or is of height >= 1, and satisfies the following properties:
  1. The root node has at least two children and one key.
  2. All nodes other than the root node have at least n and at most 2n keys.
  3. All leaf nodes are at the same level.

B-tree Operations:

- The only important operations on B-trees are search, insert, and delete.

B-tree Operations search:

- Given the search key value, x:
  1. Use any search method to search for X among keys $K_1 \ldots K_m$ of the root node.
  2. If the search was unsuccessful then three cases exist:
     a. $X < K_1$. Search node P_0.
     b. $K_i < X < K_{i+1}$, for $1 <= i < m$. Search node P_i.
     c. $X > K_m$. Search node P_m.
  3. If the pointer is null in any of the above cases, then key X is not in the tree, and the search terminates.

B-tree Operations insert:

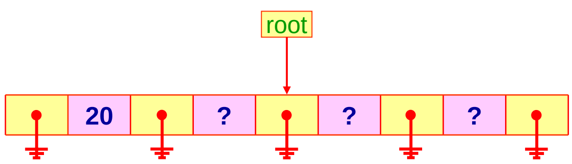
- Since all leaf nodes should be at the same level, the B-tree is forced to grow at the root:
  1. Search for the new key value, if it is not found in the tree, the search terminates at a leaf node.
  2. At this point there are two cases:
     a. If the leaf node is not full, add the new key to it.
     b. If the leaf node is full, split it into two nodes on the same level, as follows:
        i. Let S be the ordered set containing the original 2n keys and the new key.
        ii. Put the lowest n keys in the left node, and the highest n keys in the right node.
        iii. Insert the median key into the parent node.

B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.

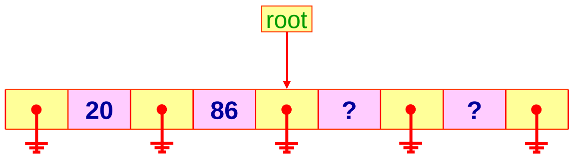
B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



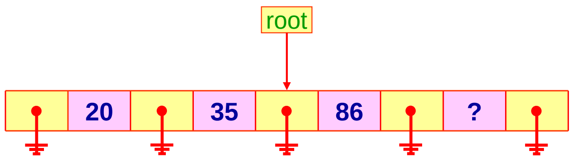
B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



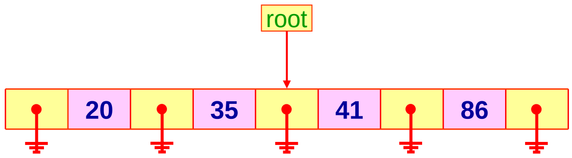
B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



B-tree Operations insert Example:

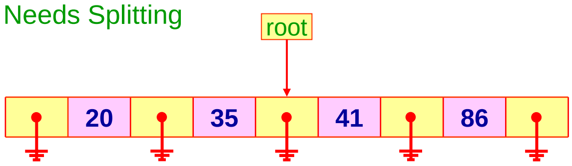
- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.

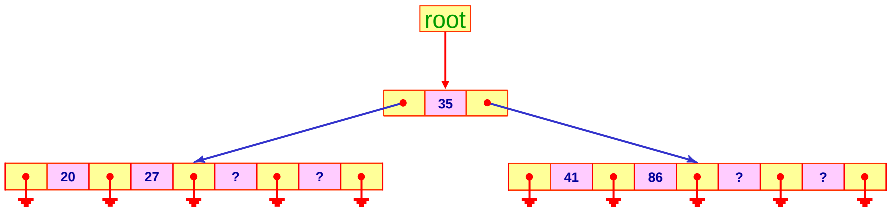
Needs Splitting



1. Form the set of **2n+1** elements: {20, 27, 35, 41, 86}
2. Choose the median element and add it to the parent node.

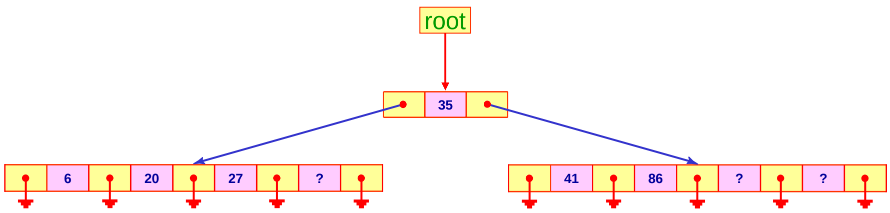
B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



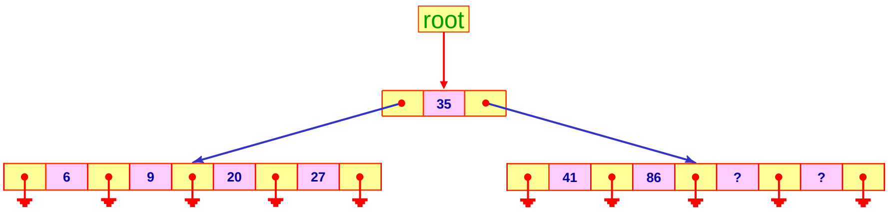
B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



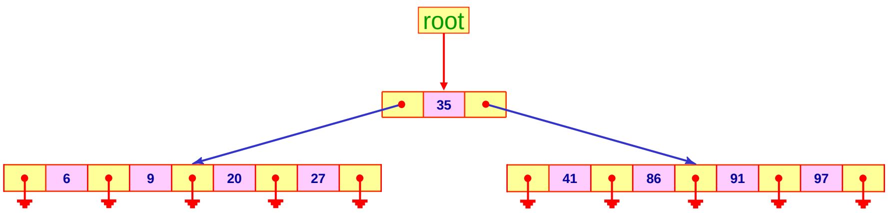
B-tree Operations
insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



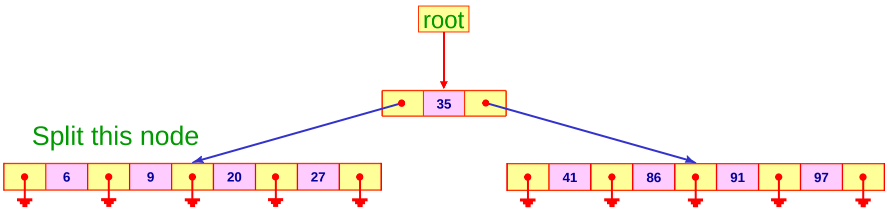
B-tree Operations
insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



B-tree Operations insert Example:

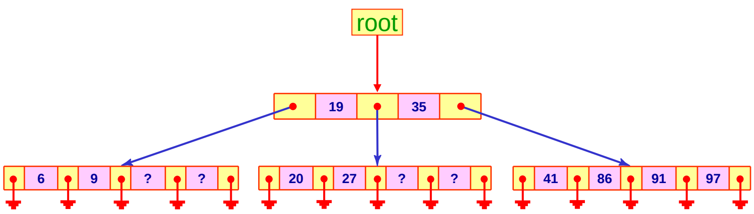
- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



1. Form the set of **2n+1** elements: {6, 9, 19, 20, 27}
2. Choose the median element and add it to the parent node.

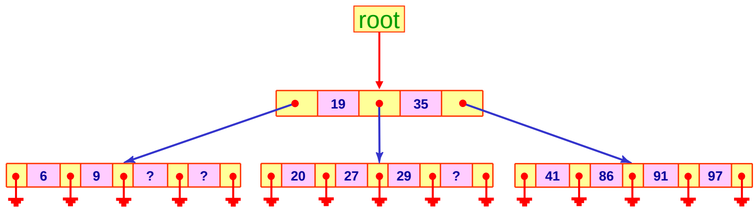
B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



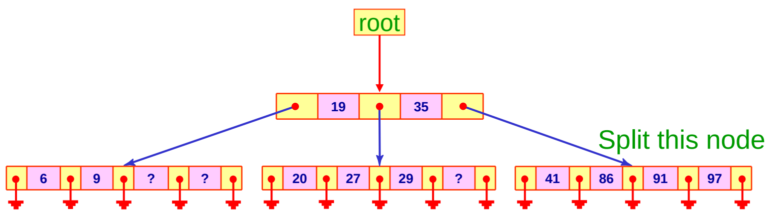
B-tree Operations
insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



B-tree Operations
insert Example:

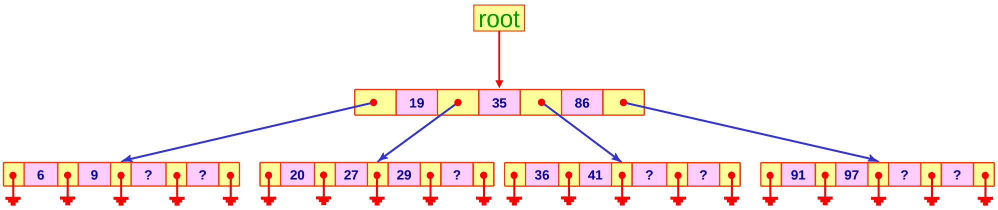
- Insert the following sequence of numbers into a B-Tree of order n=2:
20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



1. Form the set of **2n+1** elements: {36, 41, 86, 91, 97}
2. Choose the median element and add it to the parent node.

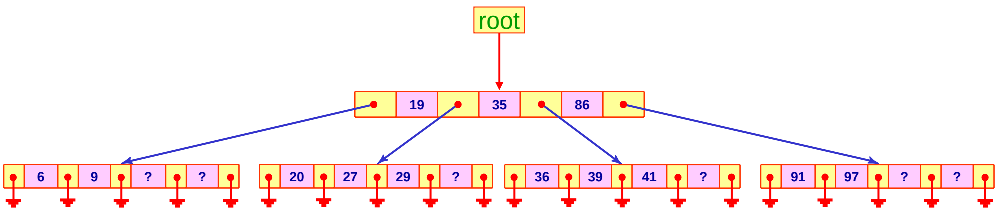
B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2: 20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



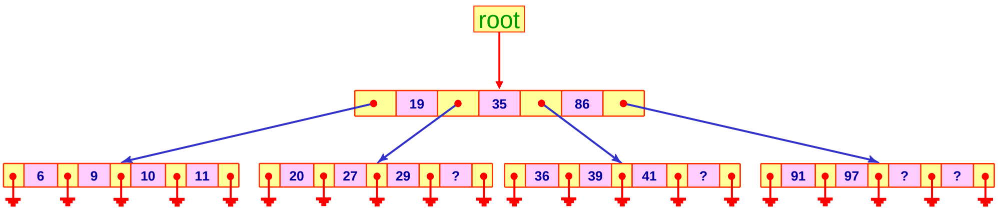
B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2: 20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



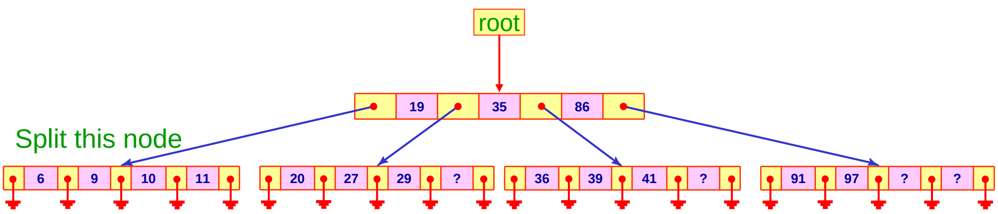
B-tree Operations
insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



B-tree Operations
insert Example:

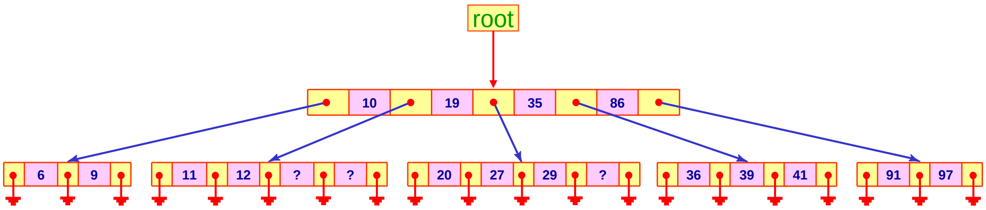
- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



1. Form the set of **2n+1** elements: {6, 9, 10, 11, 12}
2. Choose the median element and add it to the parent node.

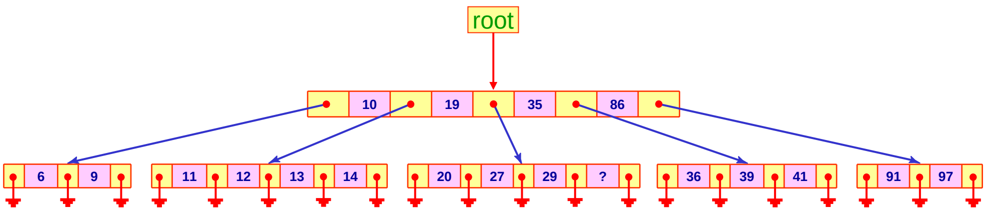
B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



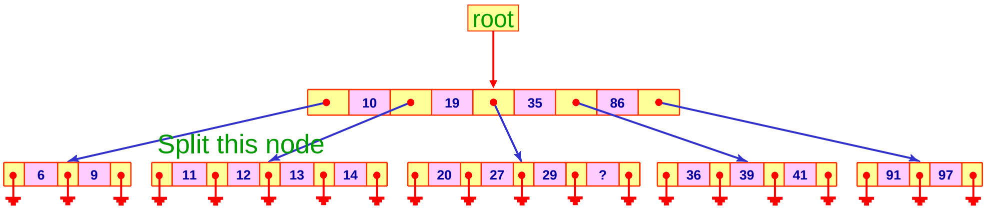
B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



B-tree Operations
insert Example:

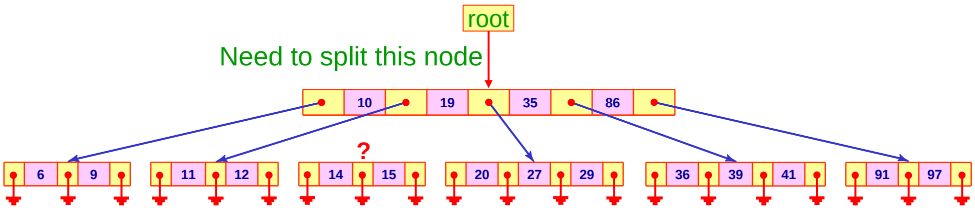
- Insert the following sequence of numbers into a B-Tree of order n=2:
20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



1. Form the set of **2n+1** elements: {11, 12, 13, 14, 15}
2. Choose the median element and add it to the parent node.

B-tree Operations
insert Example:

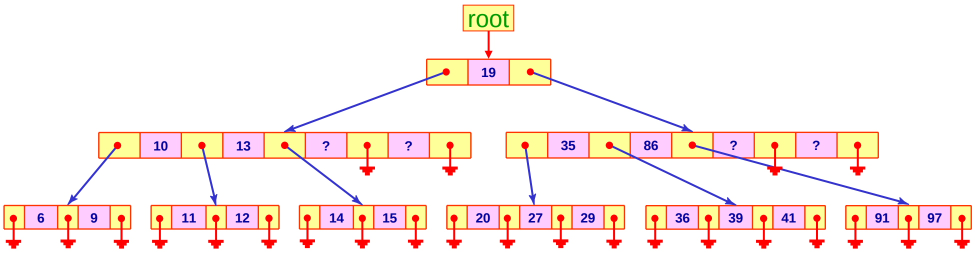
- Insert the following sequence of numbers into a B-Tree of order n=2:
20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.



1. Insert the number **13** in the parent node.
2. Form the set of **2n+1** elements: {10, 13, 19, 35, 86}
3. Choose the median element and add it to the root.

B-tree Operations insert Example:

- Insert the following sequence of numbers into a B-Tree of order n=2:
  20, 86, 35, 41, 27, 6, 9, 91, 97, 19, 29, 36, 39, 10, 11, 12, 13, 14, 15.

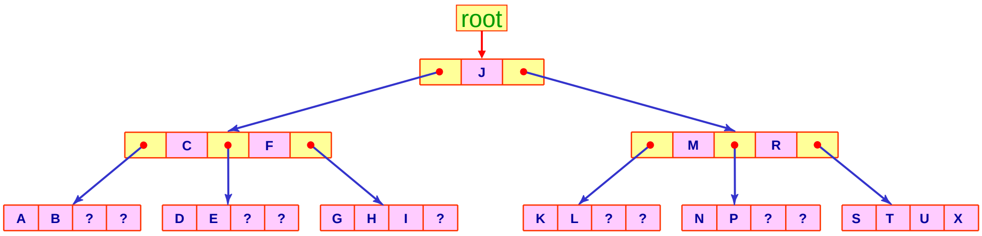


B-tree Operations delete:

- The node containing the key x to be deleted may be one of two:
  1. A leaf node. Three cases exist:
     a. The node has more than n keys. Remove key x.
     b. The node has n keys, but a neighbor has more than n keys. Remove key x and borrow one key from the neighbor.
     c. The node has n keys, and all its neighbors also have n keys. Remove key x and merge the node with its neighbor, bringing the middle key from the parent node. This is another delete propagating up.
  2. A non-leaf node. Do similar to the BST remove operation, where key x is replaced by the rightmost key in its left subtree. Then it is deleted from the leaf as in step 1.

B-tree Operations delete Examples:

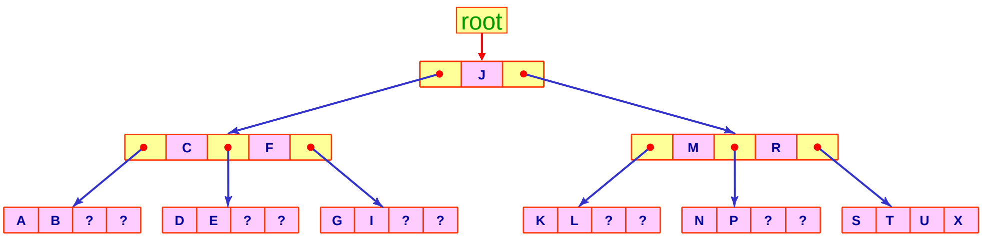
- The following is a B-Tree of order n = 2.
  1. Delete element ‘H’.



This is case 1.a : Just remove H!

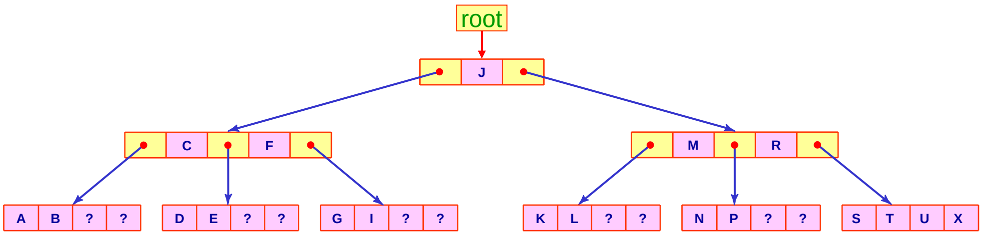
B-tree Operations delete Examples:

- The following is a B-Tree of order n = 2.
  1. Delete element ‘H’.



B-tree Operations
delete Examples:

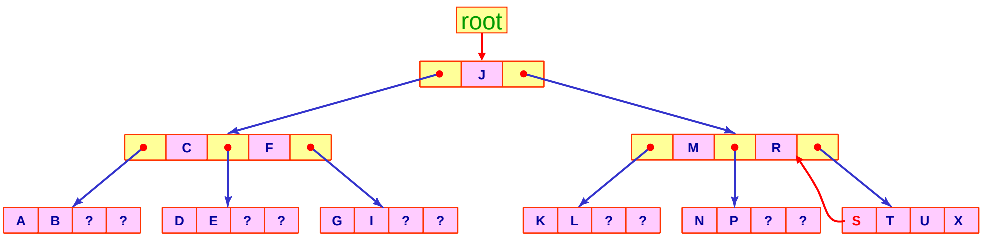
- The following is a B-Tree of order n = 2.
  2. Delete element 'R'.



This is case 2 : Replace R with the leftmost element in the subtree to its right.

B-tree Operations
delete Examples:

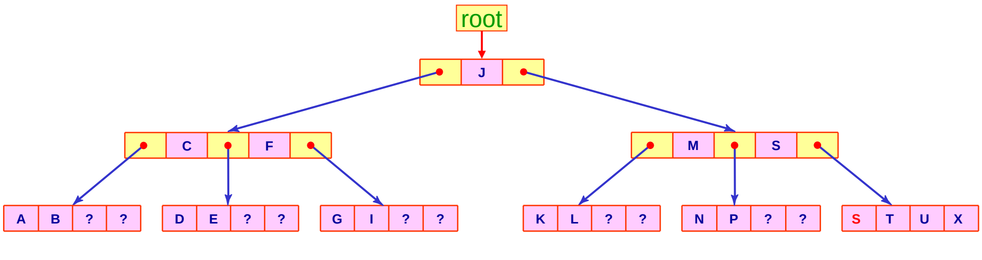
- The following is a B-Tree of order n = 2.
  2. Delete element 'R'.



This is case 2 : Replace R with the leftmost element in the subtree to its right.

B-tree Operations delete Examples:

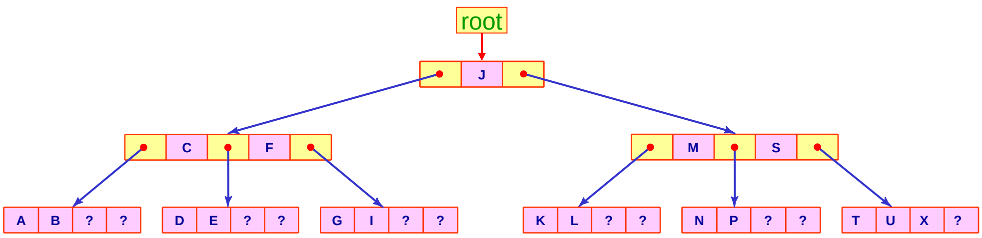
- The following is a B-Tree of order n = 2.
  2. Delete element ‘R’.



This is case 2 : Replace R with the leftmost element in the subtree to its right.
Then remove it as in case 1.a.

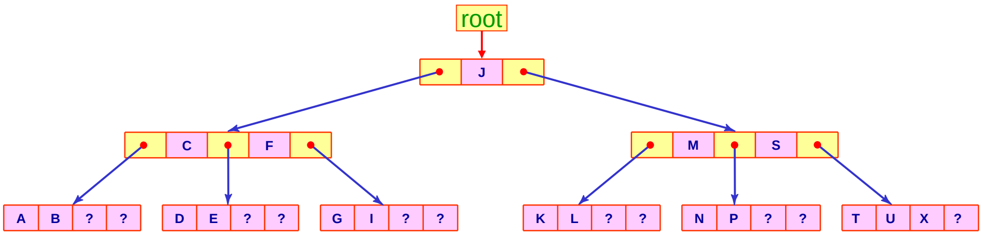
B-tree Operations delete Examples:

- The following is a B-Tree of order n = 2.
  2. Delete element ‘R’.



B-tree Operations delete Examples:

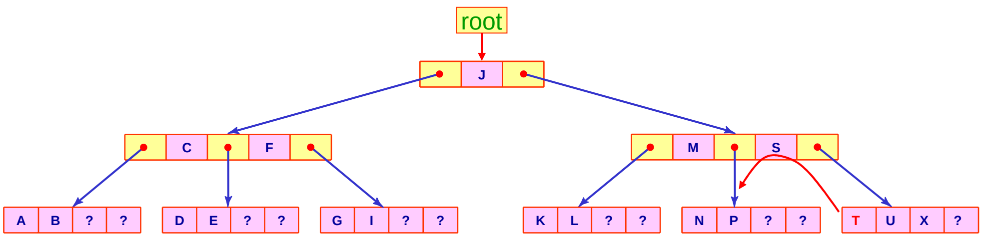
- The following is a B-Tree of order n = 2.
  3. Delete element ‘P’.



This is case 1.b : Borrow element T from the right neighbor.

B-tree Operations delete Examples:

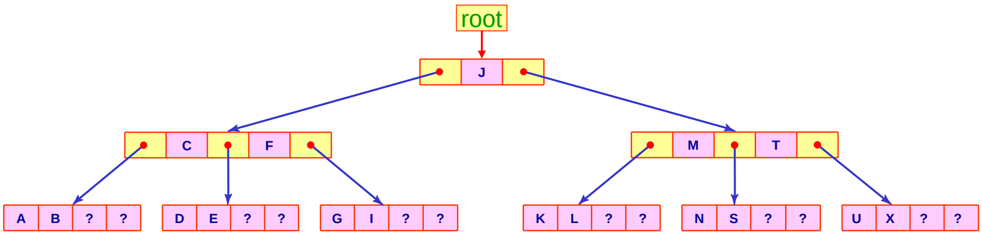
- The following is a B-Tree of order n = 2.
  3. Delete element ‘P’.



This is case 1.b : Borrow element T from the right neighbor.

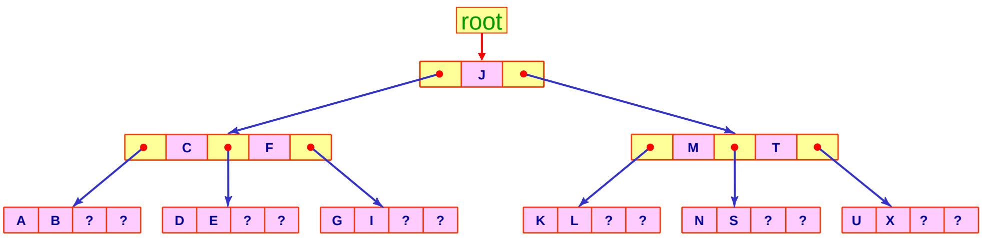
B-tree Operations delete Examples:

- The following is a B-Tree of order n = 2.
  3. Delete element ‘P’.



B-tree Operations delete Examples:

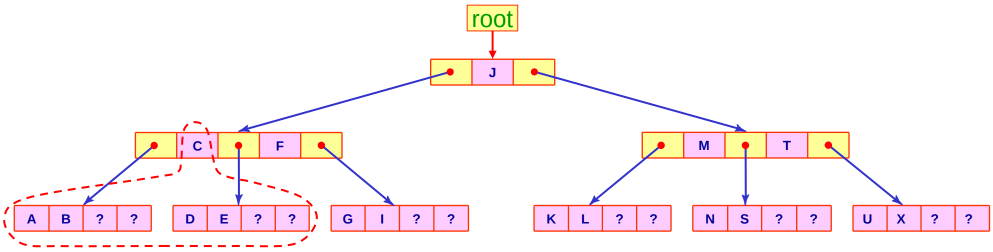
- The following is a B-Tree of order n = 2.
  4. Delete element ‘D’.



This is case 1.c : Remove element D and merge the two neighboring nodes.

B-tree Operations
delete Examples:

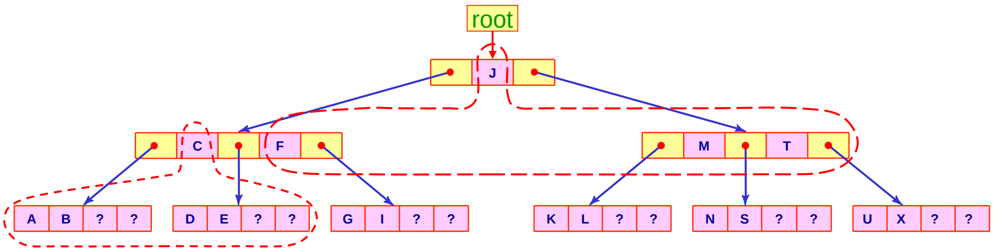
- The following is a B-Tree of order n = 2.
  4. Delete element ‘D’.



This is case 1.c : Remove element D and merge the two neighboring nodes.

B-tree Operations
delete Examples:

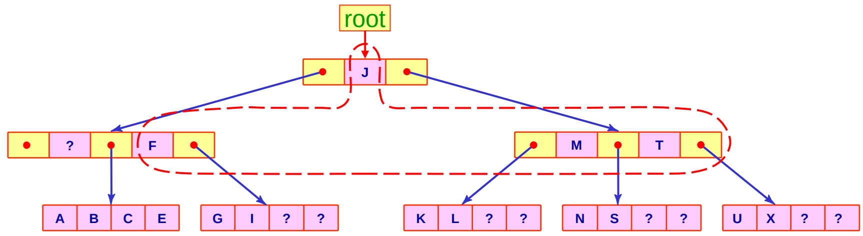
- The following is a B-Tree of order n = 2.
  4. Delete element ‘D’.



This is case 1.c : Remove element D and merge the two neighboring nodes.
This requires also merging the parent node with its neighbor!

B-tree Operations delete Examples:

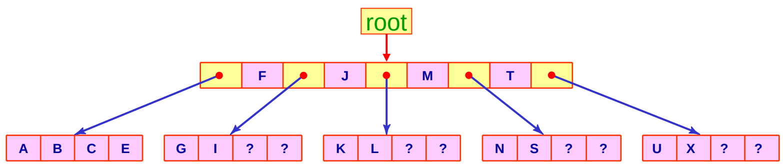
- The following is a B-Tree of order n = 2.
  4. Delete element ‘D’.



This is case 1.c : Remove element D and merge the two neighboring nodes.
This requires also merging the parent node with its neighbor!

B-tree Operations delete Examples:

- The following is a B-Tree of order n = 2.
  4. Delete element ‘D’.



B-tree Performance

- The minimum number of nodes at level $k > 1$ of a B-tree of order n is $2(n+1)^{k-2}$.
- The minimum number of keys at level $k > 1$ of a B-tree of order n is $2n(n+1)^{k-2}$.
- The maximum number of nodes at level k of a B-tree of order n is $(2n+1)^{k-1}$.
- The maximum number of keys at level k of a B-tree of order n is $2n(2n+1)^{k-1}$.

B-tree Performance (cont.)

- In the worst case, a B-tree of order n containing d nodes has a height, $h = O(\log_n d)$.

  For example, if $n = 100$, and $d = 10^7$, then $h = 4$.

- The search operation requires $O(\log_n d)$ secondary storage accesses.
- Space utilization is at least 50%.

B-tree Performance (cont.)

- In the worst case, the insert and delete operations take a total of $3h+1$ page accesses as follows:
  - h for the search,
  - $2h$ for the splitting or merging, and
  - One for storing the updated node.
- The average case for both is $h+1$.
- For most practical applications, a good choice for n is 100 ➔ 200, giving $h_{max} \cong 4$.

B-tree Applications:

- B-trees are widely used for file systems and databases:
  - Windows: NTFS.
  - Mac: HFS, HFS+.
  - Linux: XFS, Ext3, JFS.
  - OS/2: HPFS.
  - Databases: ORACLE, DB2, SQL, INGRES, PostgreSQL, … etc.
- Usually, the node size, M = the page size.