DATA STRUCTURES & ALGORITHMS

Trees : M ways

Instructor: Engr. Laraib Siddiqui

M-Way Search Tree

It can have more than two children.

M is called the *degree* of the tree.

Requirement????

Favors retrieval and manipulation of data stored in external memory.

Goal???

Minimizes the accesses while retrieving a key from a file.

M-Way Search Tree

It can have m pointers and m-1 keys in one node.

Properties:
- Each node has 0 .. m subtrees
- A node with k < m subtrees, contains k-1 keys.
- The key values of the first subtree are all less than the key value.
- The data entries are ordered.

Representation:

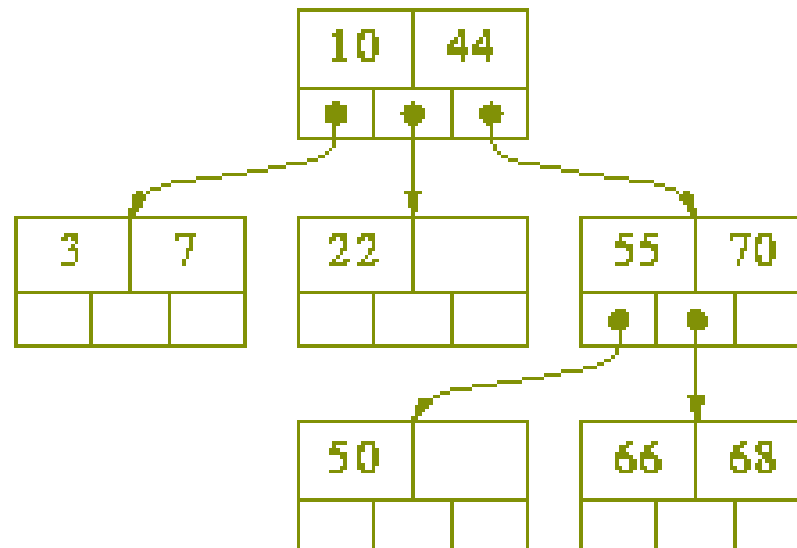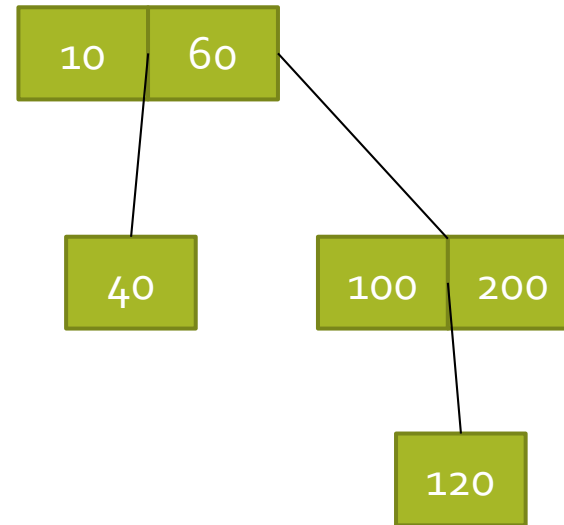| P_0	K_1	P_1	K_2	P_2	....	K_n	P_n

Where,
P are pointers to the subtree.
K are key values in the node

3-way search tree
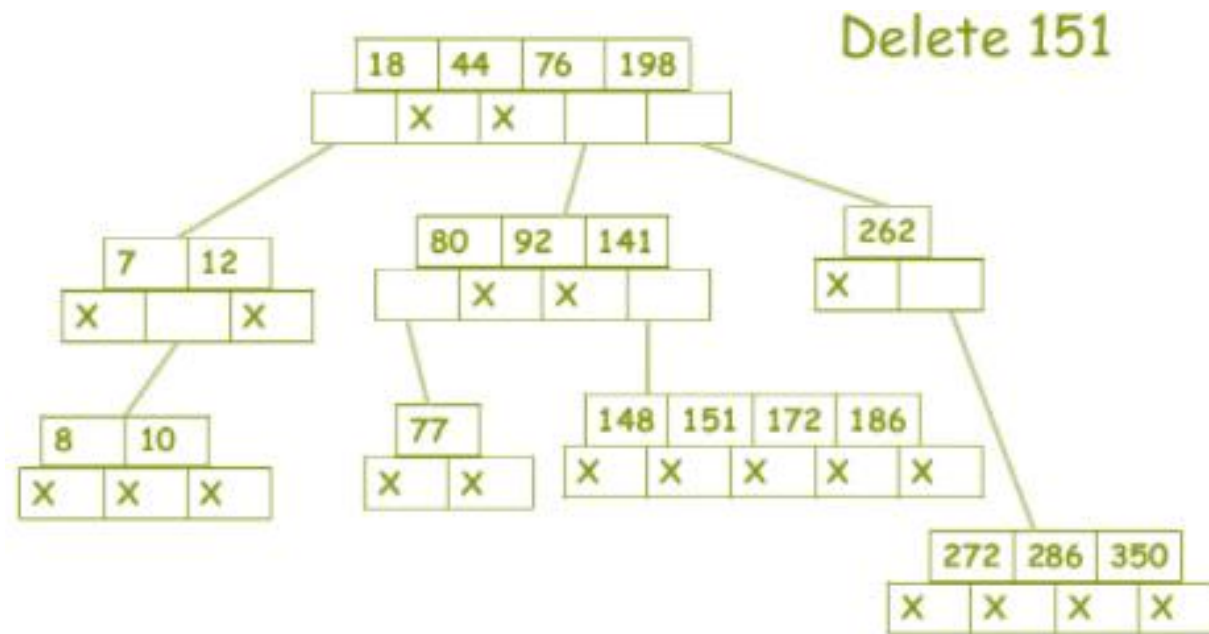
3-way search tree

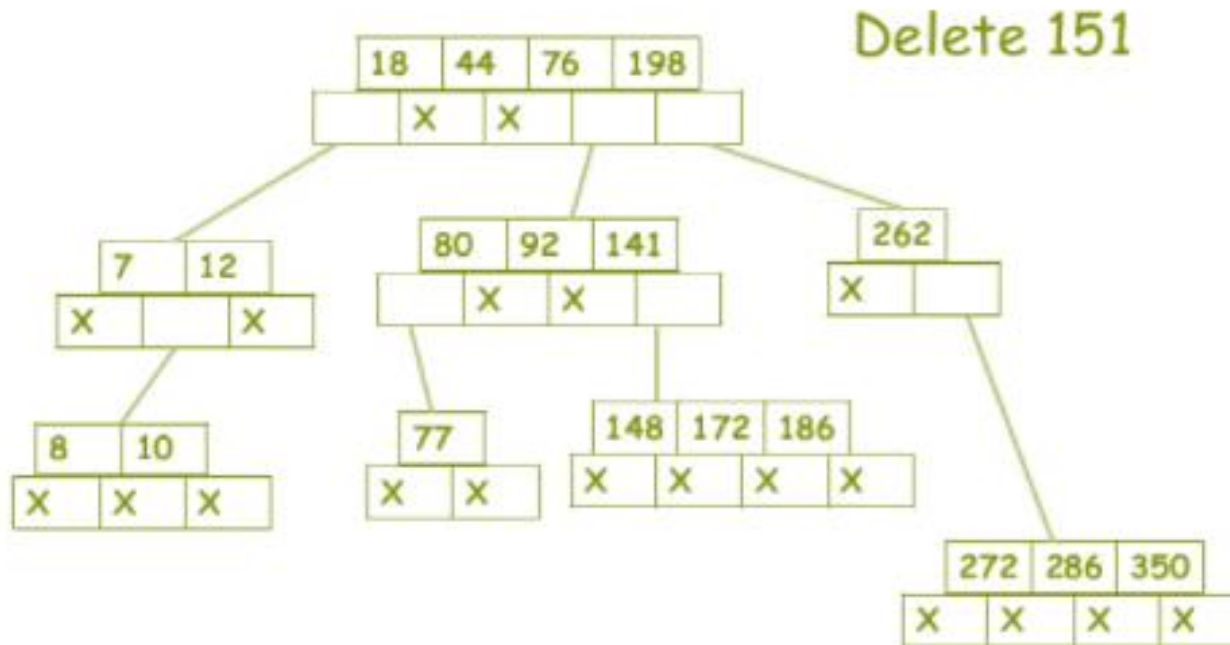Insert 10, 60, 100, 200,40,120

4-way search tree

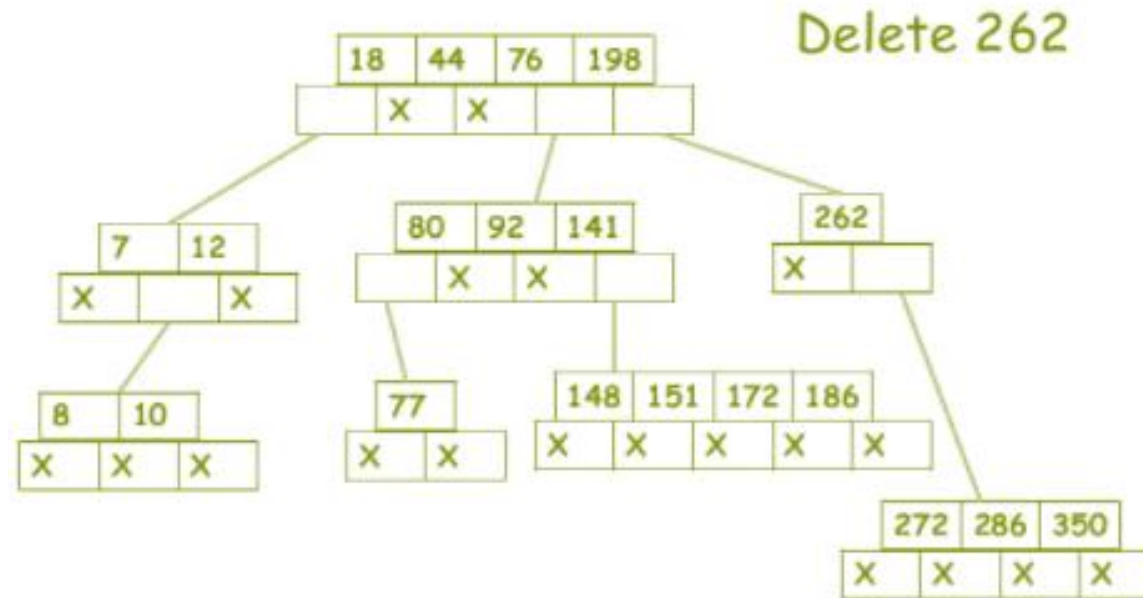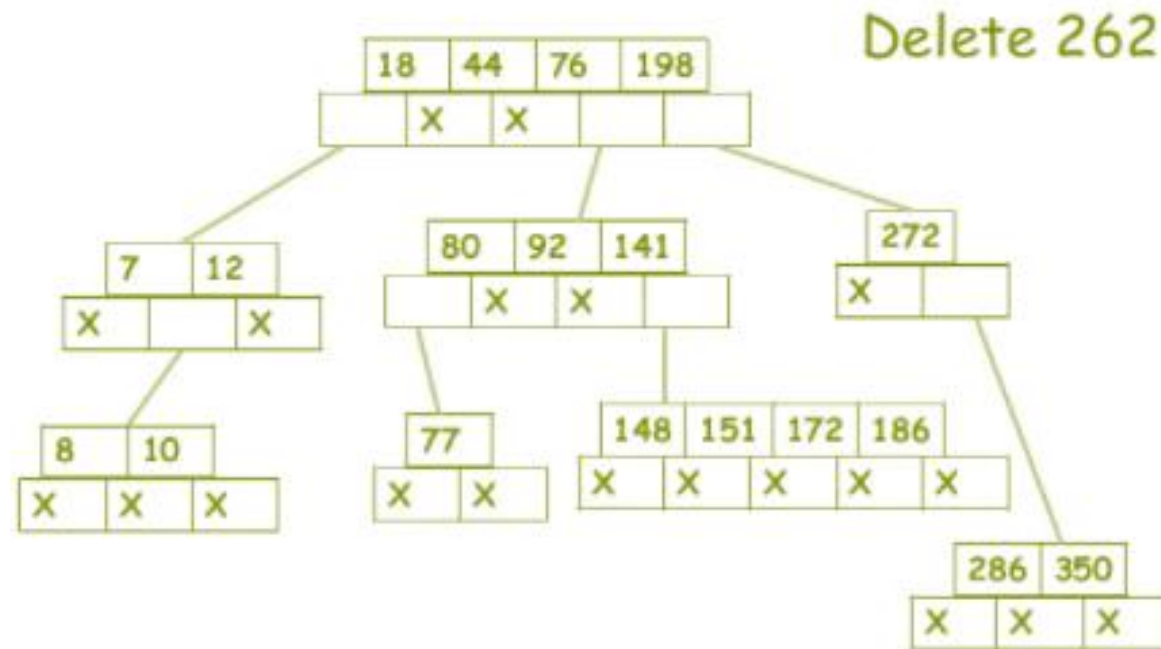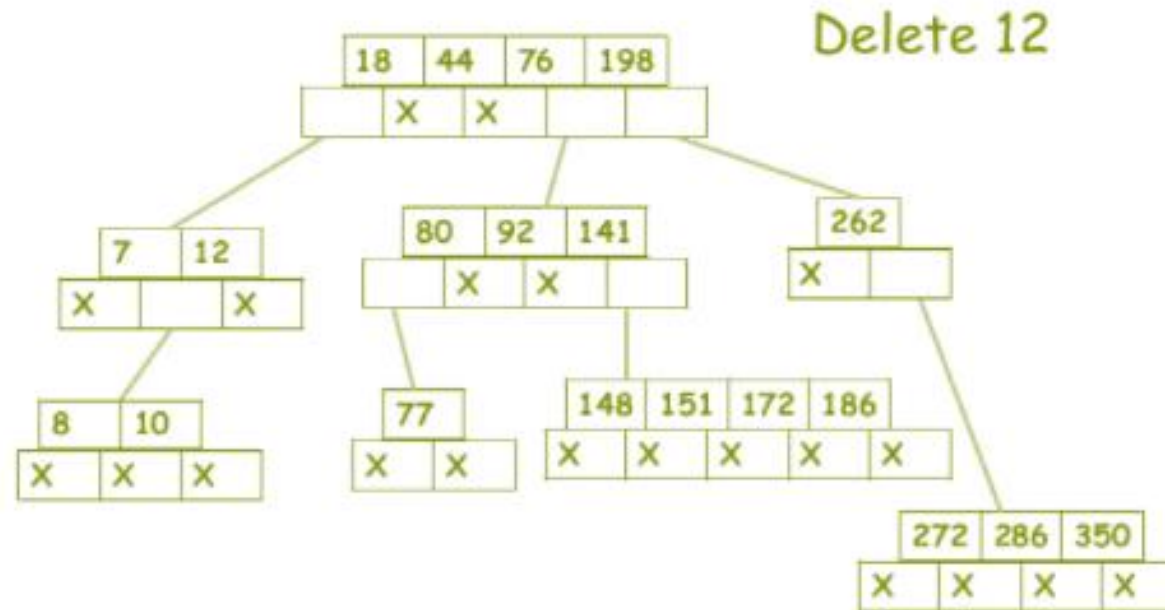Insert 10, 60, 100, 200,40,120

?

5 Way - Deletion

Delete 151

5 Way - Deletion

Delete 151

5 Way - Deletion

Delete 262

18	44	76	198
	X	X	

7 12 / X X

80 92 141 / X X

262 / X

8 10 / X X X

77 / X X

148 151 172 186 / X X X X X

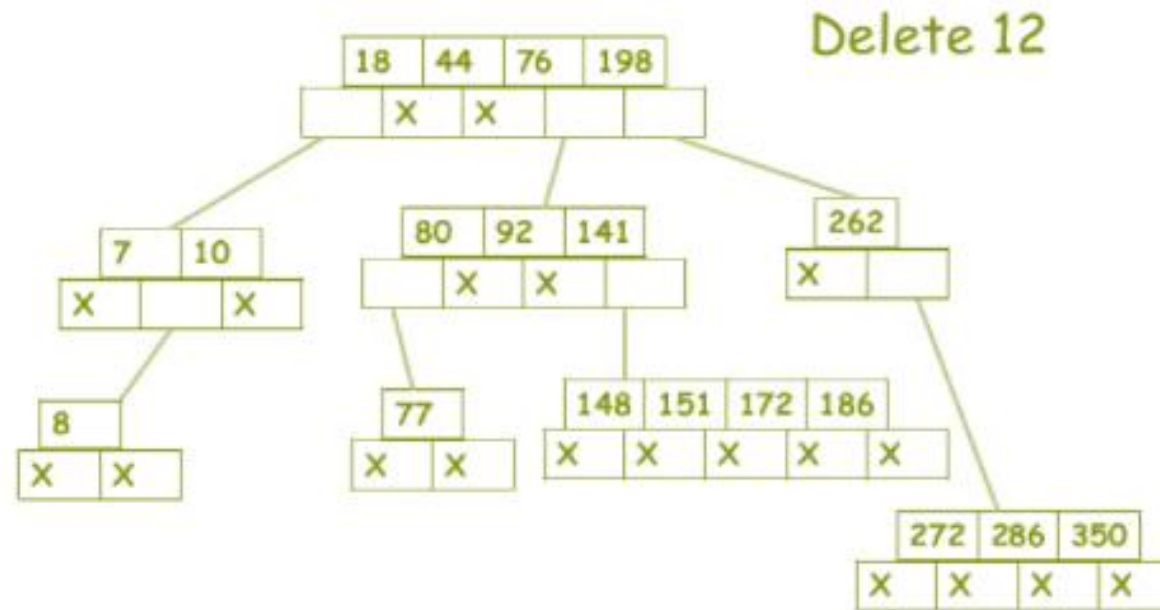272 286 350 / X X X X

5 Way - Deletion

Delete 262

5 Way - Deletion

5 Way - Deletion

Delete 12

B Tree

B Tree is a specialized m-way tree that can be widely used for disk access.

B tree of order m contains all the properties of an M way tree. In addition, it contains the following properties.

- Every node in a B-Tree contains at most m children.
- Every node in a B-Tree except the root node and the leaf node contain at least [m/2] children.
- The root nodes must have at least 2 nodes.
- All leaf nodes must be at the same level.
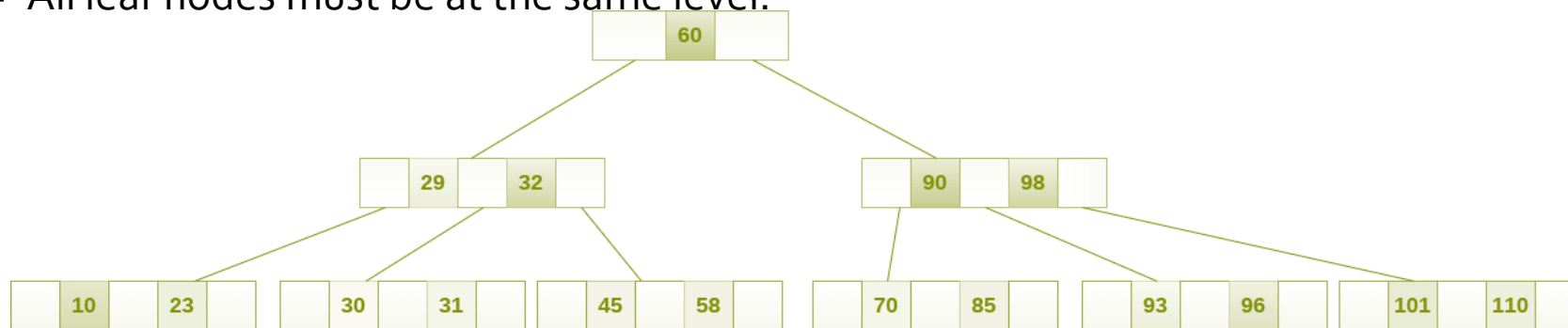
B tree order 3
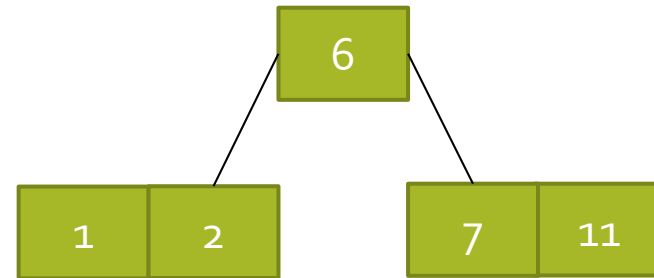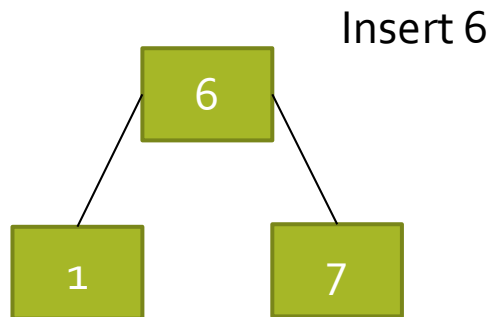
Insert 1, 7, 6, 2, 11, 4, 8

Insert 6

```
    1  |  7
```

Insert 1, 7

```
        6
       / \
      1   7
```

```
          6
         / \
    1 | 2   7 | 11
```

Insert 2, 11

B Tree- Insertion

1. Traverse the B Tree in order to find the appropriate leaf node at which the node can be inserted.

2. If the leaf node contain less than m-1 keys then insert the element in the increasing order.

3. Else, if the leaf node contains m-1 keys, then follow the following steps.
   a) Insert the new element in the increasing order of elements.
   b) Split the node into the two nodes at the median.
   c) Push the median element upto its parent node.
   d) If the parent node also contain m-1 number of keys, then split it too by following the same steps.
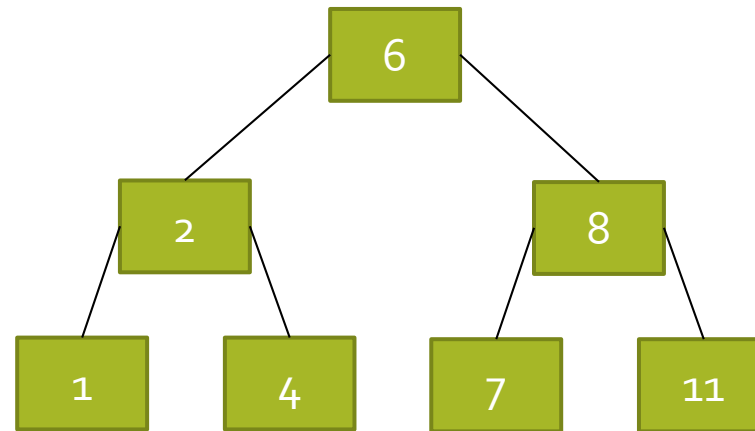
Insertion- B tree order 3

Insert 4

Insert 8

16
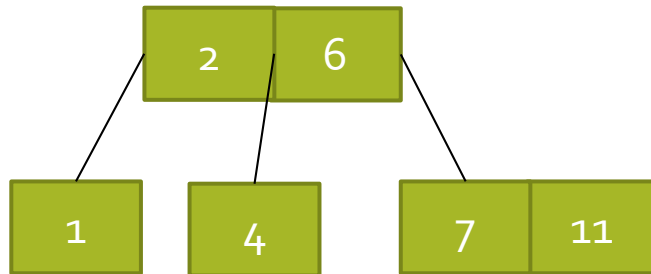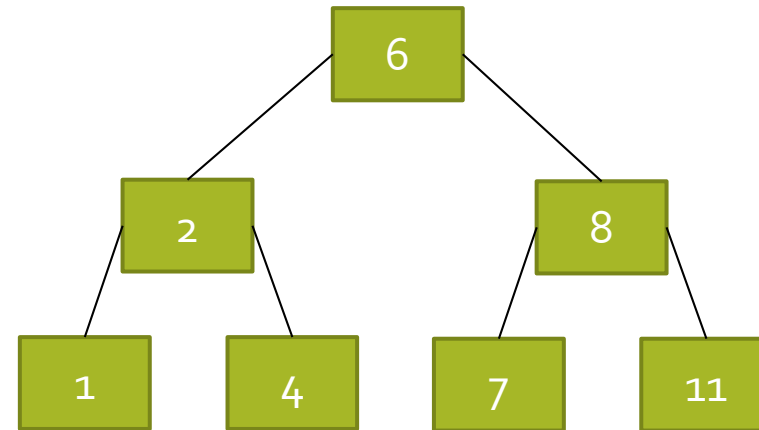
Insertion- B tree order 3
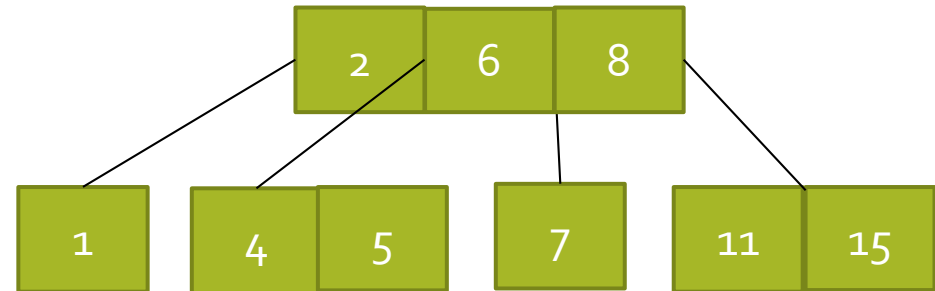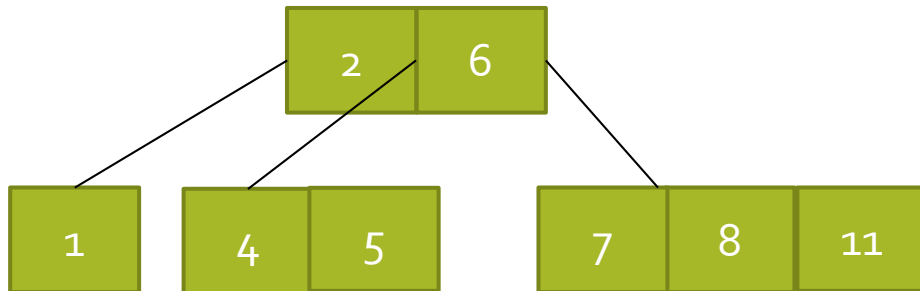

Insert 5, 15, 3, 12 ????

Insertion- B tree order 4

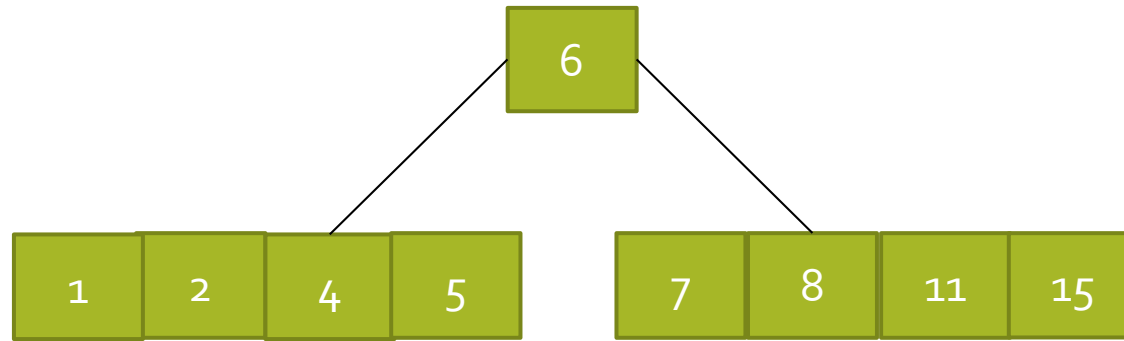Insert 1, 7, 6, 2, 11, 4, 8, 5, 15

Insertion- B tree order 4

Insert 1, 7, 6, 2, 11, 4, 8, 5, 15

Insertion- B tree order 5

Insert 1, 7, 6, 2, 11, 4, 8, 5, 15

B tree- Deletion

Cases

- **Leaf Node**
  - More than min no of keys - Delete
  - Min no of keys – Borrow from sibling, If sibling is at min no of keys, then merge with parent.

Internal node
  - Successor will replace
  - If not, then borrow from sibling
  - If not, then merge with parent

Min Children: **[m/2]**
Max Children: **m**
Min Keys: **[m/2] - 1**
Max Keys:**m-1**

B tree- Deletion

1. Locate the node.

2. If there are more than [m/2] - 1 keys in the node then delete the desired key from the node.

3. If the node doesn't contain [m/2] - 1 keys then complete the keys by taking the element from right or left sibling.
    a) If the left sibling contains more than [m/2] - 1 elements then push its largest element up to its parent and move the intervening element down to the node where the key is deleted.
    b) If the right sibling contains more than [m/2] - 1 elements then push its smallest element up to the parent and move intervening element down to the node where the key is deleted.

4. If neither of the sibling contain more than [m/2] - 1 elements then create a new node by joining two leaf nodes and the intervening element of the parent node.

5. If parent is left with less than [m/2] - 1 nodes then, apply the above process on the parent too.
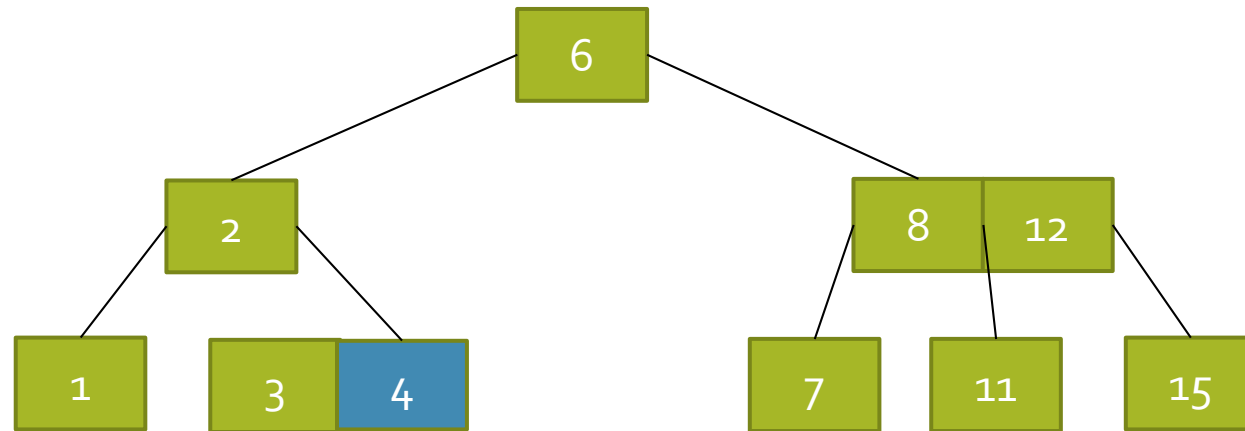
Deletion
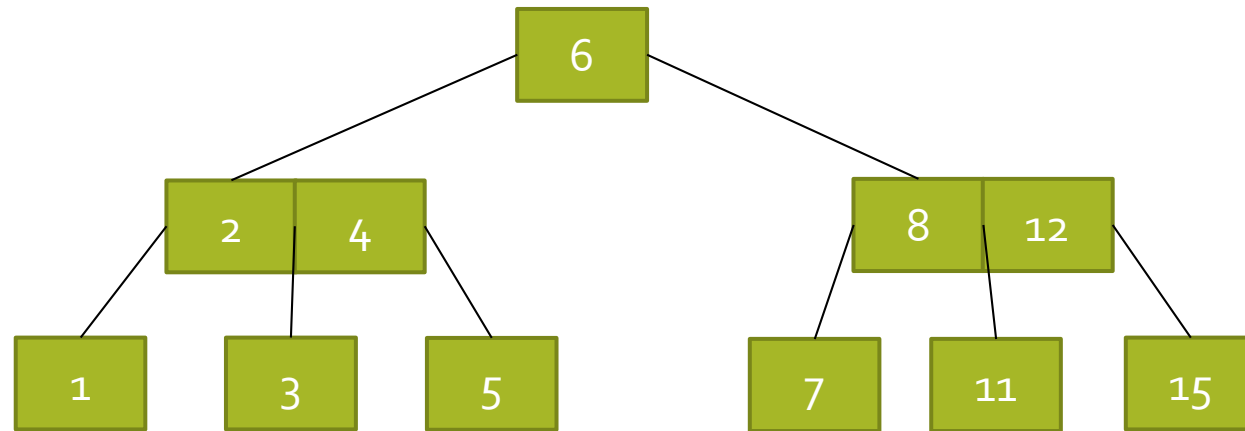
Delete 5

For order 3, min keys can be ⌈m/2⌉ - 1
=> ⌈3/2⌉ - 1
= > 1

Merge node

Deletion

Delete 12

Deletion
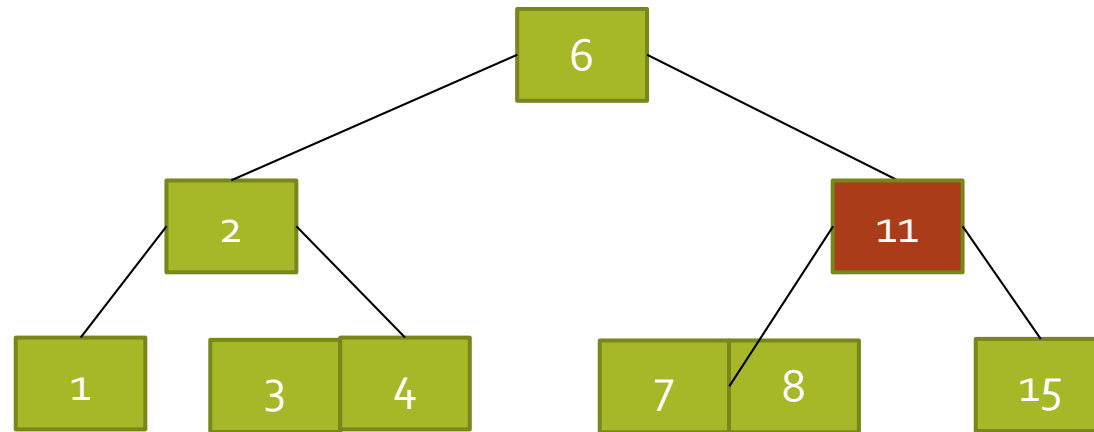
Complexity

	B Tree
Access	O(Log(n))
Search	O(Log(n))
Insertion	O(Log(n))
Deletion	O(Log(n))