CSE-281: Data Structures and Algorithms
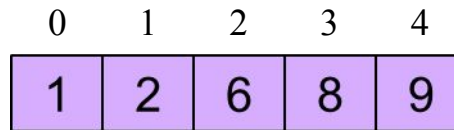
Trees
(Chapter-7)

Eftekhar Hossain
Lecturer
Dept. of ETE, CUET

CUET

Topics to be Covered

- Binary Tree
- Tree Traversal
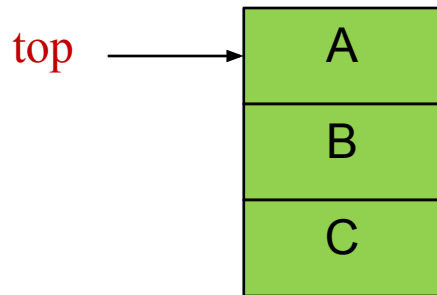- Binary Search Tree
- AVL Tree
- B-Tree

Introduction to Trees

 Linear Data Structure

	0	1	2	3	4
	1	2	6	8	9

Array

| 23 | 5000 | → | 44 | 4750 | → | 87 | 3970 |

Linked List

top →
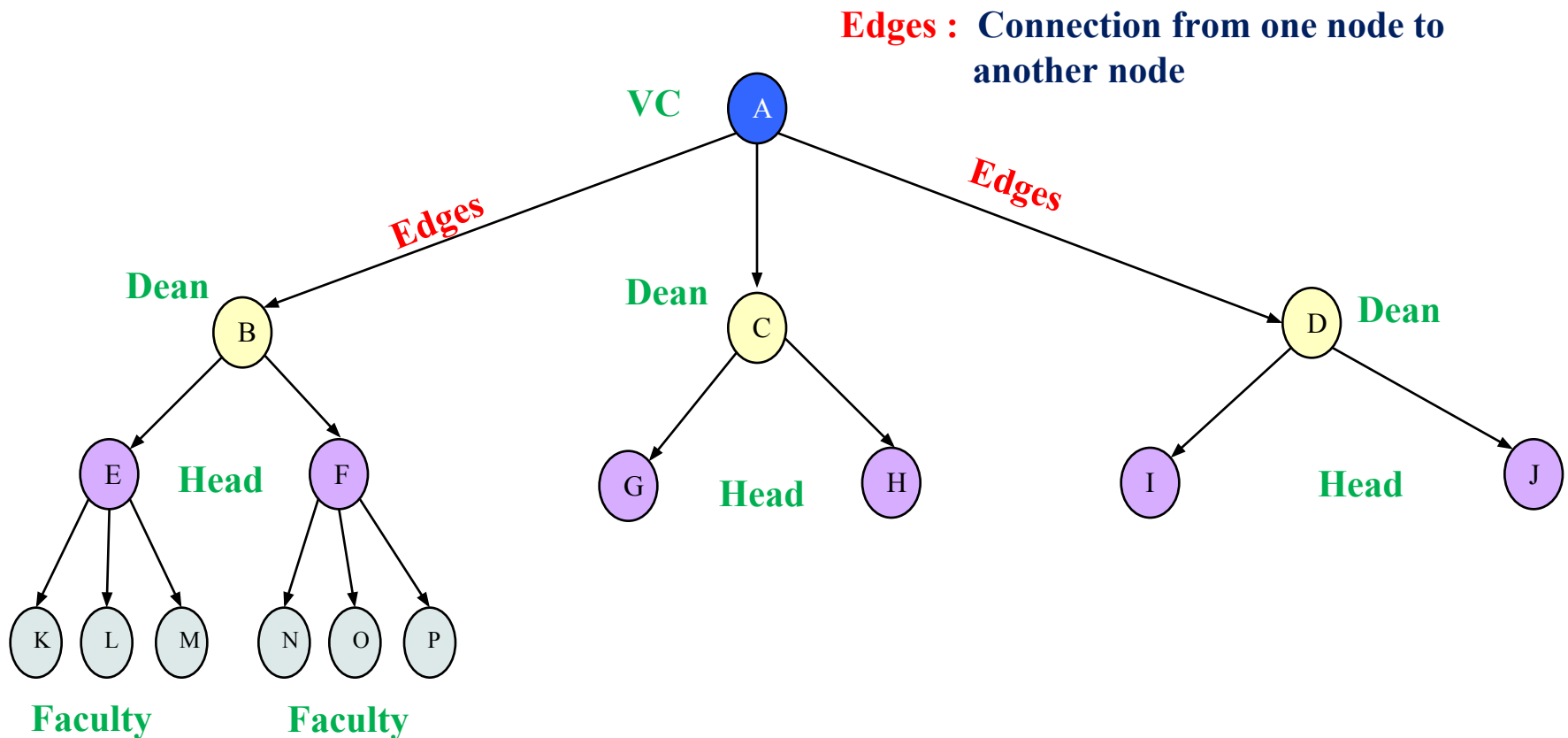
A
B
C

Stack

| 2 | 3 | 8 |

FRONT   REAR
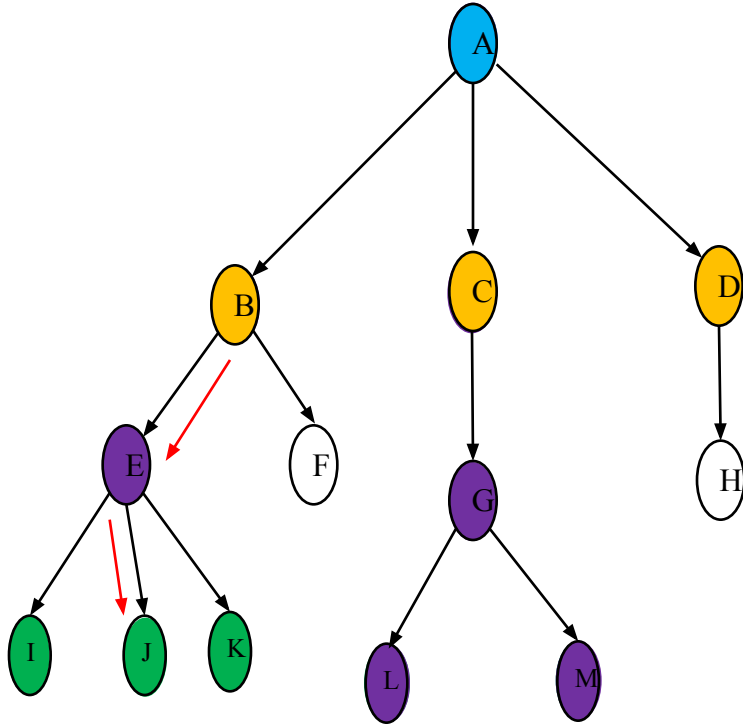
Queue

3

Introduction to Trees

- Tree can be defined as a collection of entities (nodes) linked together to simulate a hierarchy.

Edges : **Connection from one node to another node**

VC A

Edges

Edges

Dean B

Dean C

D Dean

E Head

F

G Head

H

I Head

J

K L M

N O P

Faculty   **Faculty**

Tree Terminologies

Nodes: A , B, C, D, E, F, G, H ,I ,J, K, L, M

Root: A

Parent Node: B is parent of E & F

Child Node: L & M are the children of G

Leaf Nodes: (External Nodes)   doesn't have any children ☐ I , J, K, L, M

Non – Leaf nodes :have at least one children A , B, C, D, E, G

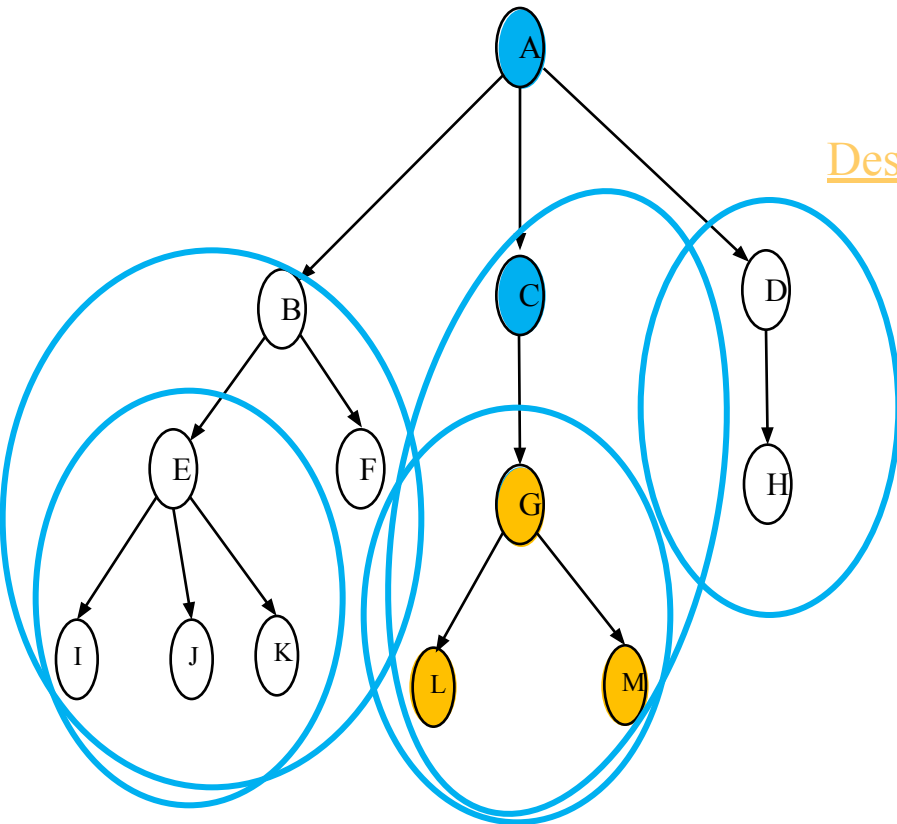Path :   sequence of consecutive edges from source node to destination node

B ☐ J

B ☐ E   E☐ J

Tree Terminologies

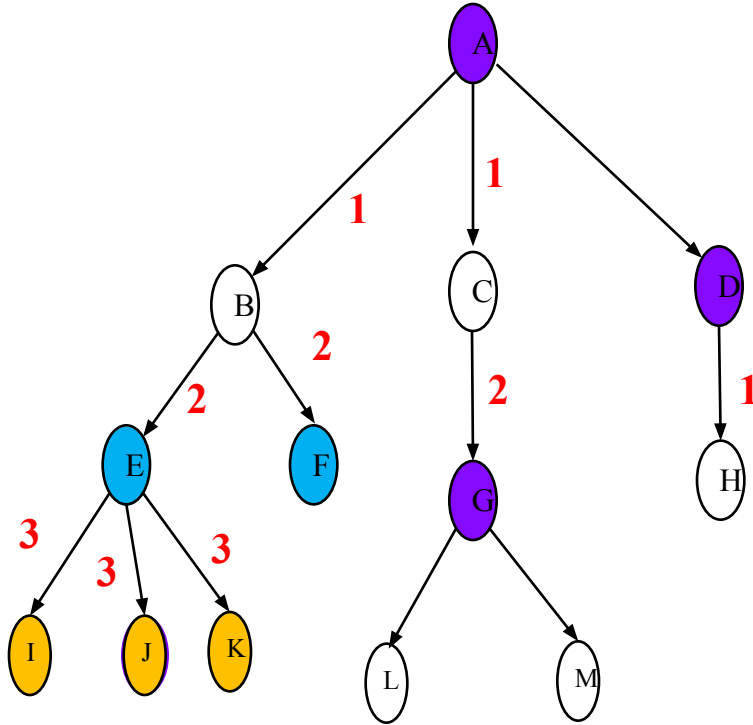Ancestor : any predecessor node on the path from root to that node .

Ancestor of L □ A, C, G

Descendent : any successor node on the path from that node to the leaf node .

Descendent of C □ G, L, M

Sub Tree of a Tree T

Tree Terminologies

Siblings: children's of same node

Degree : degree of any node is the number of children that any node have

degree of any leaf node is 0

degree of a Tree is the maximum number of degree any node have

Degree of this Tree is 3

Height and Depth of a node
May or may not be same

Height of a tree is equal to the Height of Root A

Depth : length of path from root to that node

Depth of node G ☐2 , J☐3

Height : no. of edges in the longest path from that node to a leaf node.
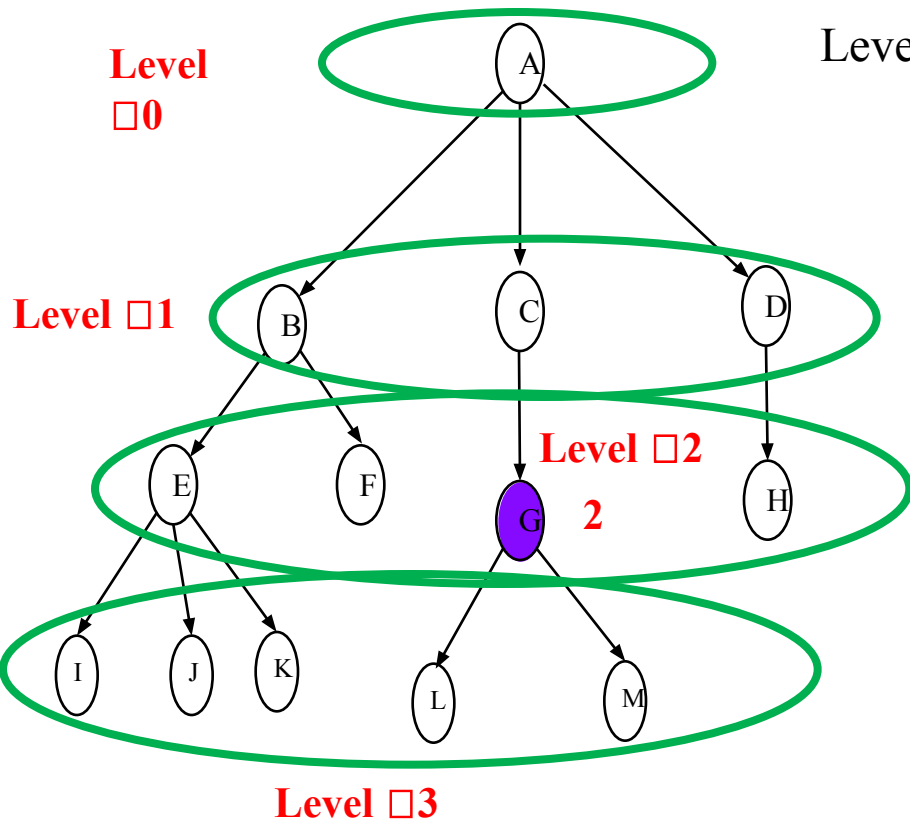
Height of node D ☐1 , A☐3

Tree Terminologies

Level of a node : distance from root to that node

Level of G $\Box$ 2

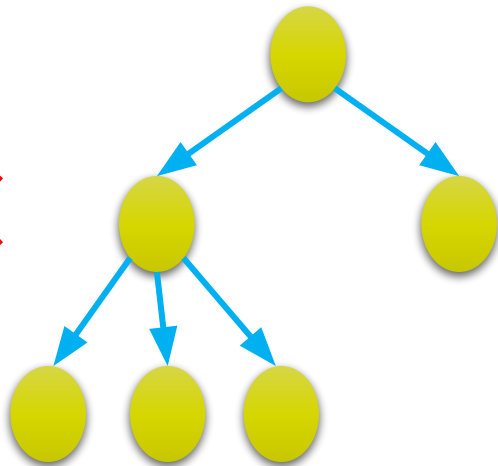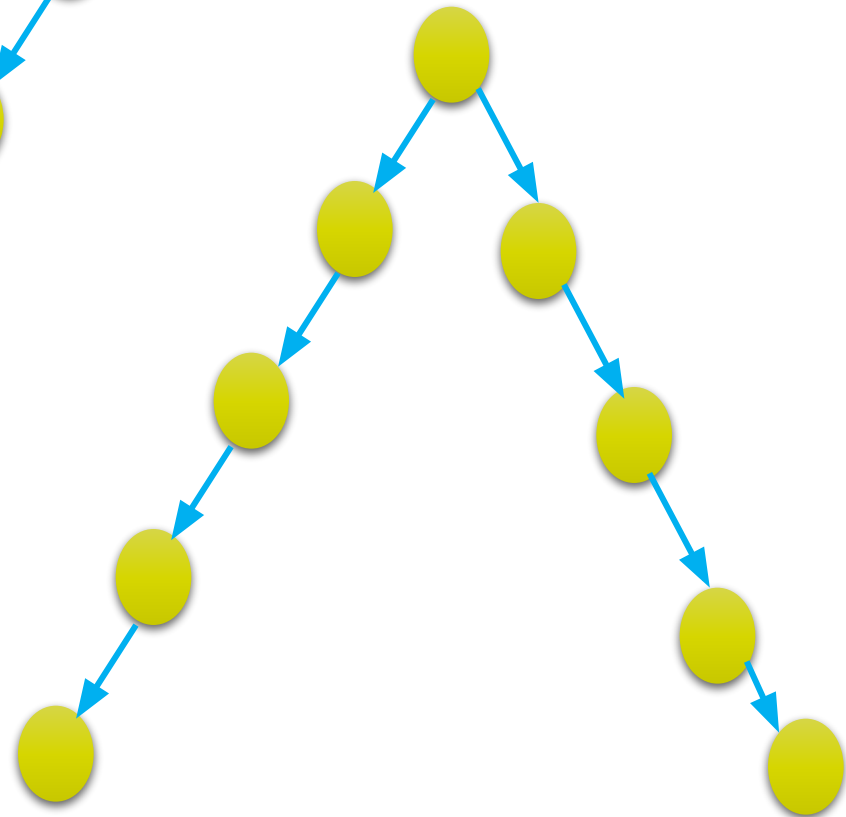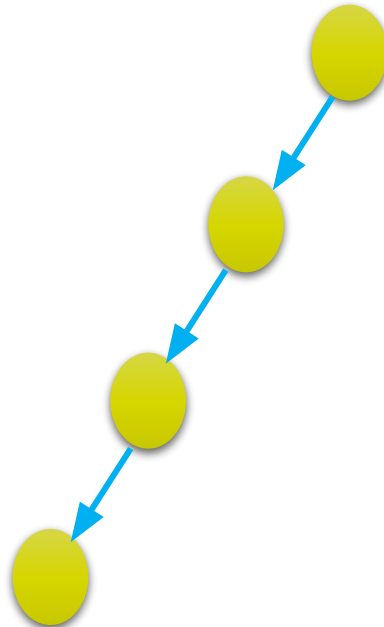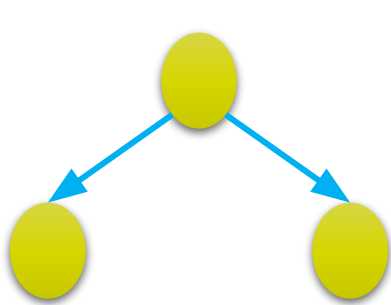Level of a Tree is equal to the Height of the Tree , here it is 3

Level of a node is equal to the depth of a node

If a Tree have n number of nodes then there must have $n-1$ edges

Level $\Box$0

Level $\Box$1

Level $\Box$2

2

Level $\Box$3

A

B   C   D

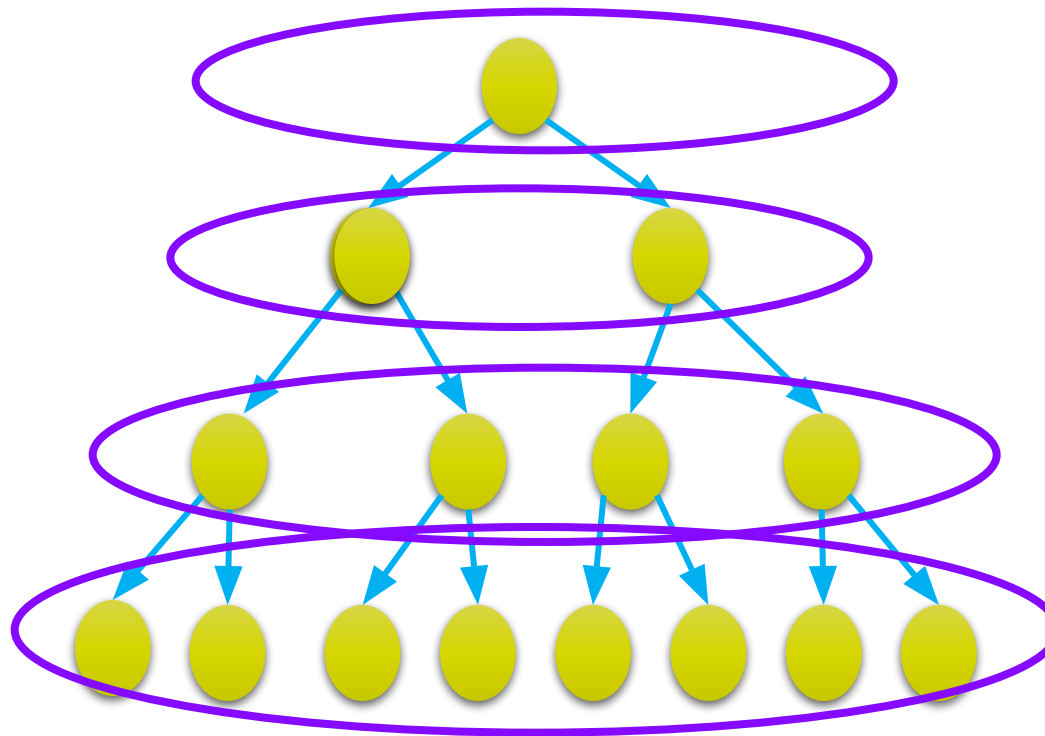E   F   G   H

I   J   K   L   M

Binary Trees

☐ Each node have at most 2 children

Properties of Binary Trees

- If a Tree have n number of nodes then there must have $n - 1$ edges

Level	Max Nodes
0	1
1	2
2	4
3	8

Properties of Binary Trees

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

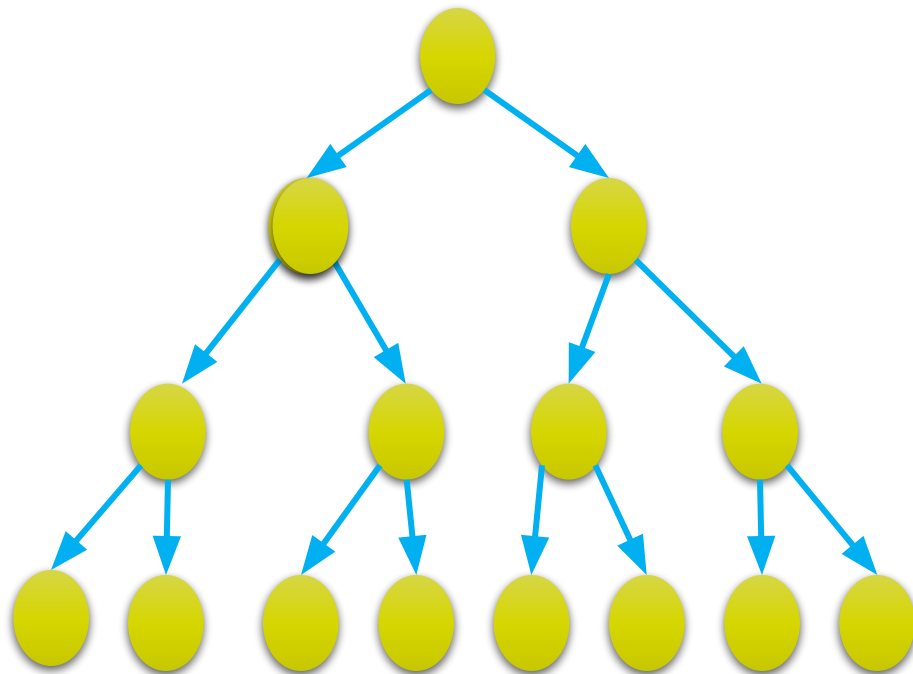| Level	Max Nodes
0	1
1	2
2	4
3	8

If a Tree have n number of nodes then there must have n – 1 edges

If a Tree have n number of nodes then there must have n – 1 edges

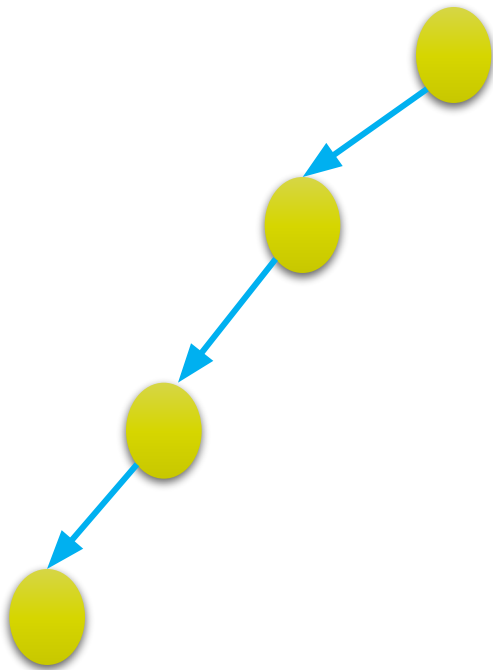If a Tree have n number of nodes then there must have n – 1 edges

If a Tree have n number of nodes then there must have n – 1 edges

Properties of Binary Trees

If a Tree have n number of nodes then there must have $n-1$ edges

Level	Min Nodes
0	1
1	1
2	1
3	1

Properties of Binary Trees

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges
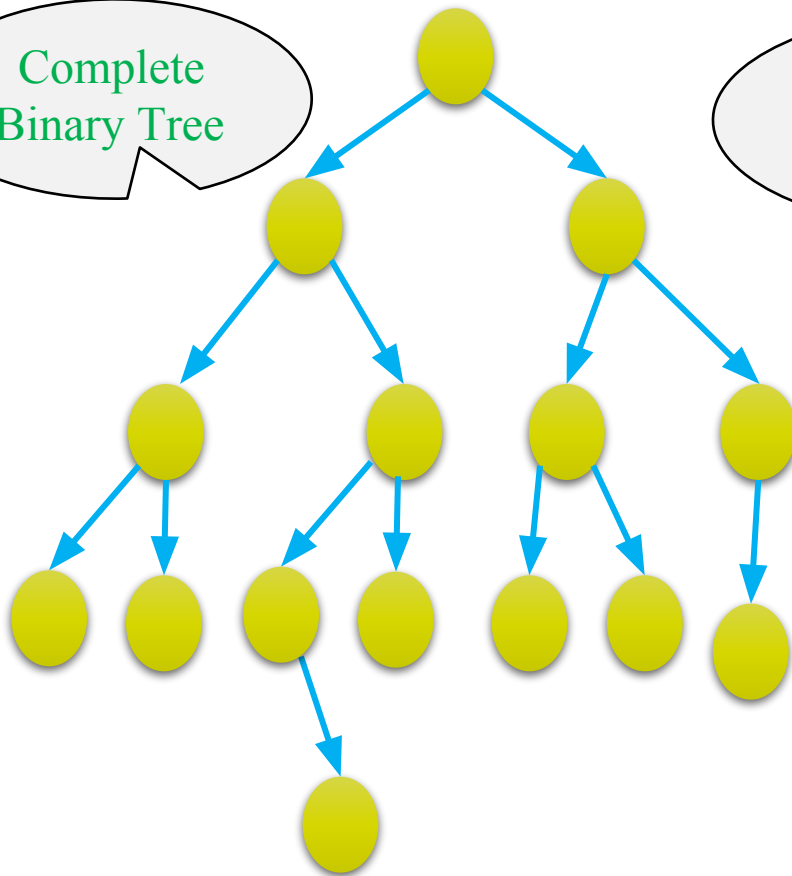
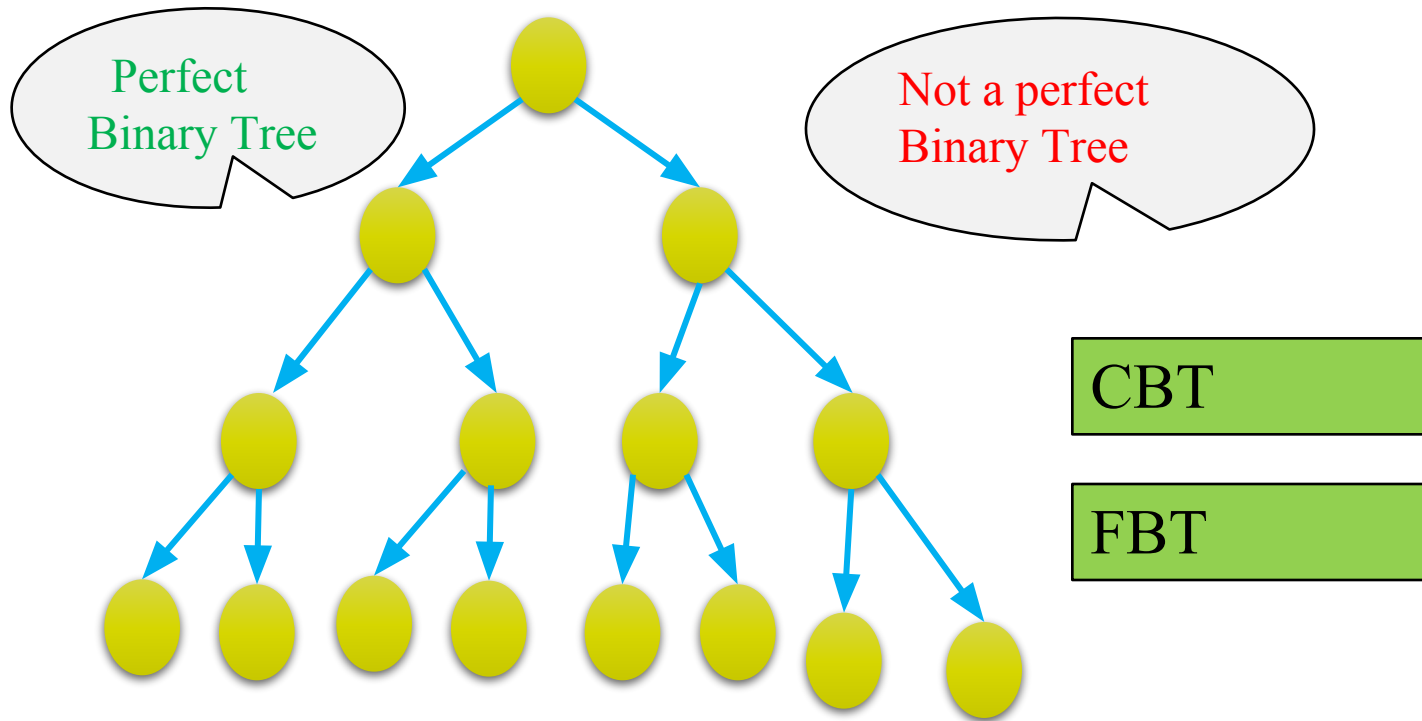If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

13

Types of Binary Trees

Full / Proper / Strict

Complete

Perfect Binary Tree

Degenerate Binary Tree

14

Full Binary Tree

Each node have either 0 or 2 children

Each node is contain exactly 2 children's except leaf nodes

Full Binary Tree

Not a Full Binary Tree

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

15

Complete Binary Trees

All levels are completely filled (except possibly the last level)

And the last level must have nodes as left as possible

Complete Binary Tree
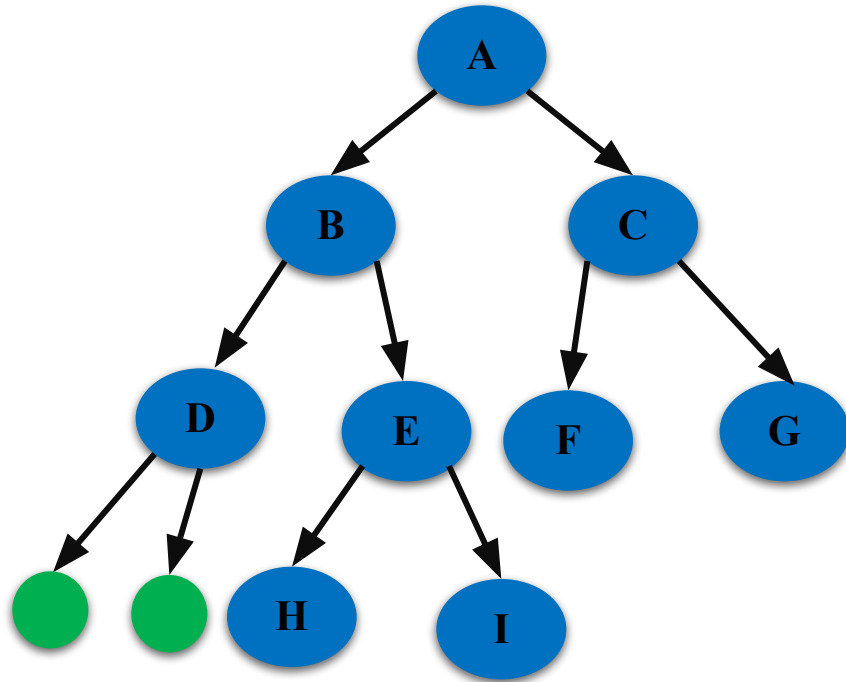
Not a complete Binary Tree

If a Tree have n number of nodes then there must have $n - 1$ edges

If a Tree have n number of nodes then there must have $n - 1$ edges

If a Tree have n number of nodes then there must have $n - 1$ edges

If a Tree have n number of nodes then there must have $n - 1$ edges

Perfect Binary Trees

All internal nodes have 2 children

And all leaf nodes are at same level

Perfect Binary Tree

Not a perfect Binary Tree

CBT

FBT

A PBT can be a CBT or FBT or both

Degenerate Binary Trees

All internal nodes have only one children

Left skewed

Right skewed
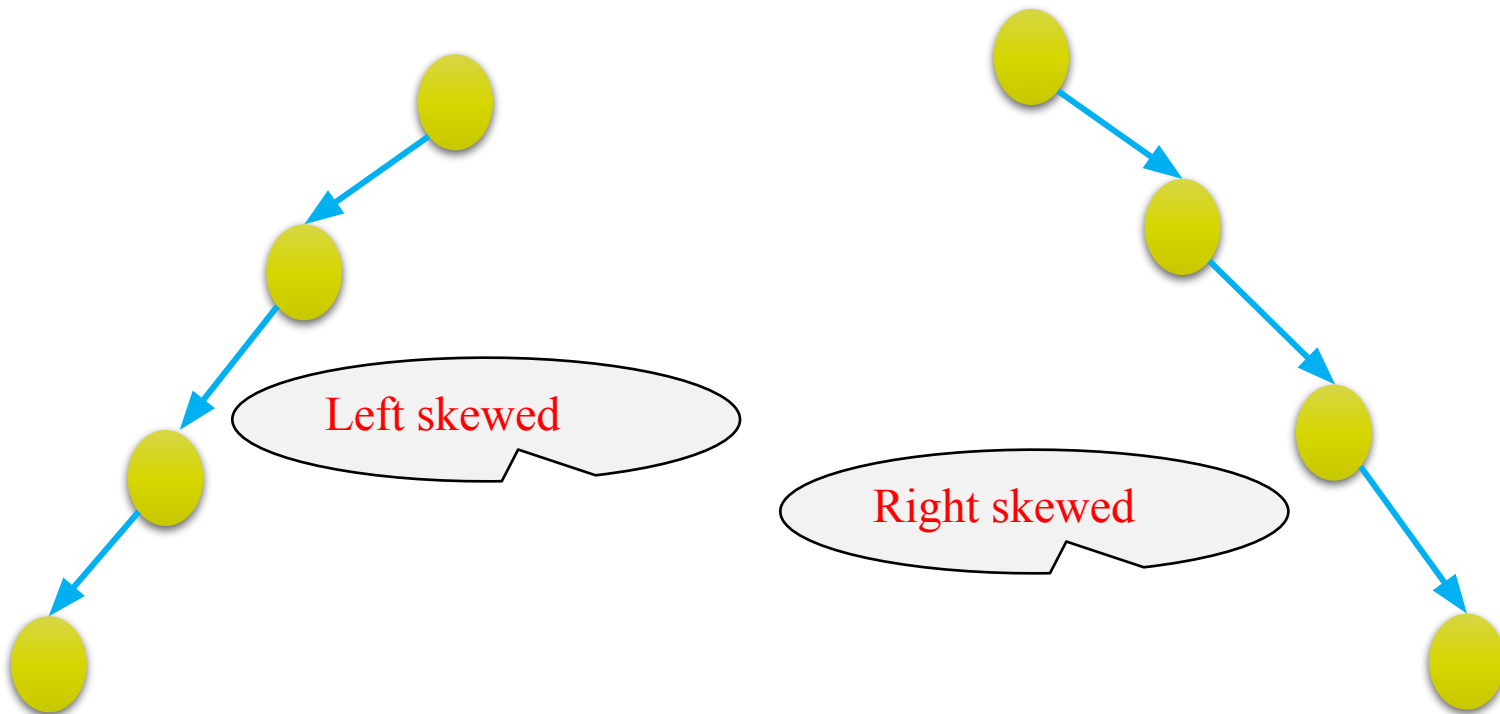
Array Representation of BT
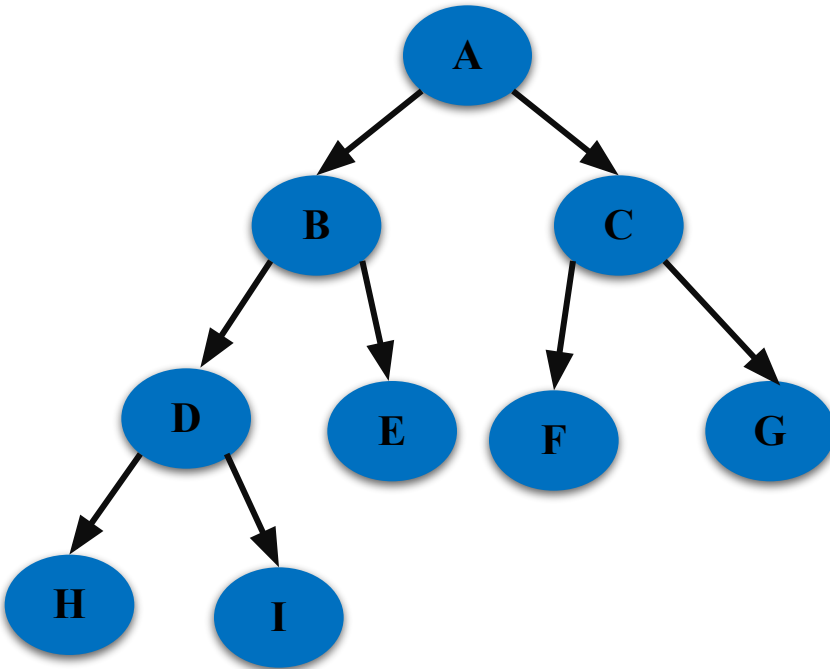
Fill up the array using the level
Of the tree from left to right

How to find the parent child relation
From the array representation ?

0	1	2	3	4	5	6	7	8
A	B	C	D	E	F	G	H	I
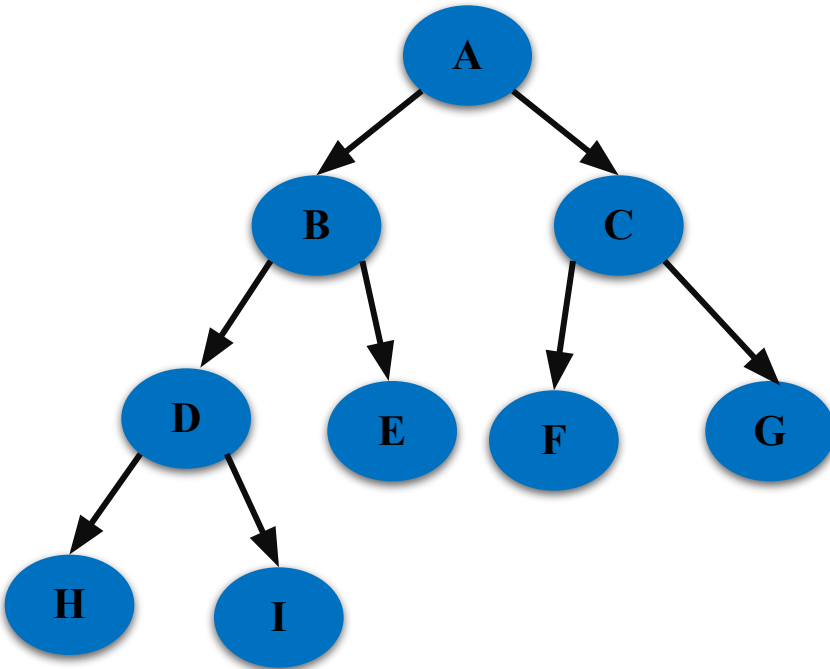
Array Representation of BT

If a Tree have n number of nodes then there must have $n - 1$ edges

If a Tree have n number of nodes then there must have $n - 1$ edges

If a Tree have n number of nodes then there must have $n - 1$ edges

If a Tree have n number of nodes then there must have $n - 1$ edges

0	1	2	3	4	5	6	7	8
A	B	C	D	E	F	G	H	I

Array Representation of BT

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

If a Tree have n number of nodes then there must have $n-1$ edges

For array representation a Tree have to be a Complete Binary Tree

0	1	2	3	4	5	6	7	8
A	B	C	D	E	F	G	H	I

0	1	2	3	4	5	6	7	8	9	10
A	B	C	D	E	F	G			H	I

Array Representation of BT

0	1	2	3	4	5	6
A	-	B	-	-	-	C
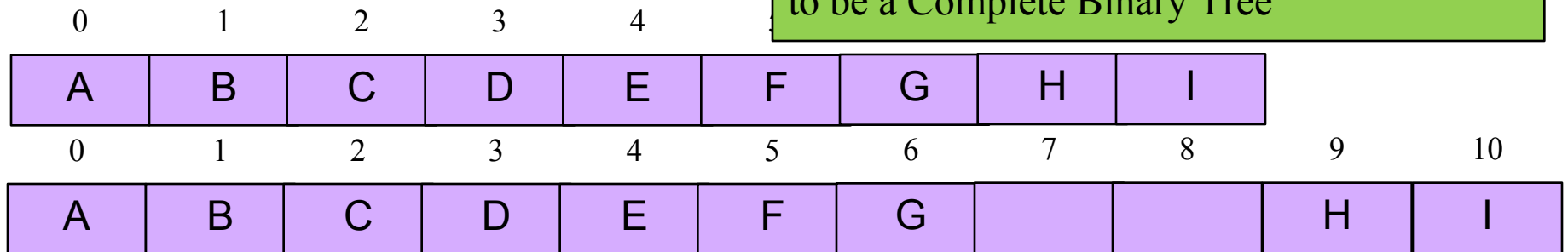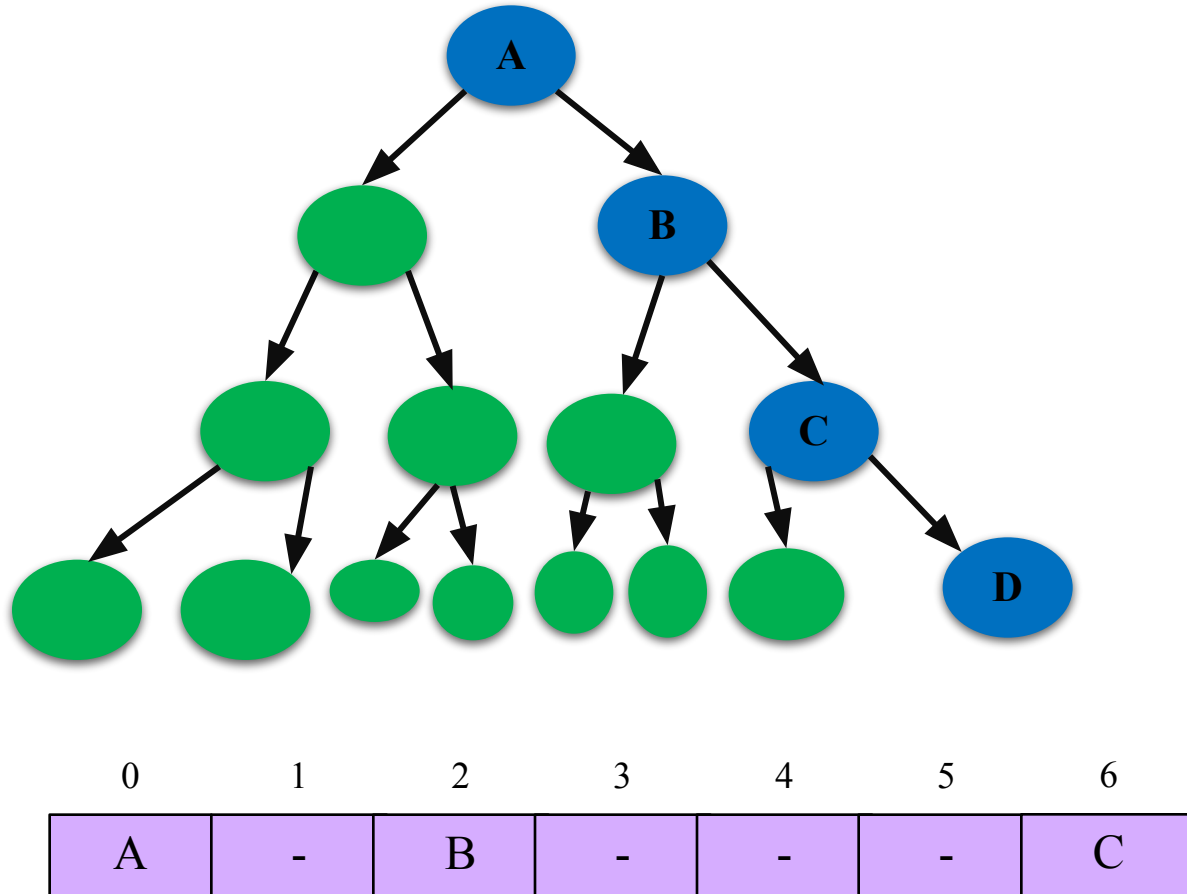
Tree Traversal

Processing or Reading the data of a node exactly once in some order in a tree.

Inorder: Left Root Right

Preorder: Root Left Right

Postorder: Left Right Root

Inorder Traversal

Inorder: Left Root Right

B	D	A	G	E	C	H	F	I

24
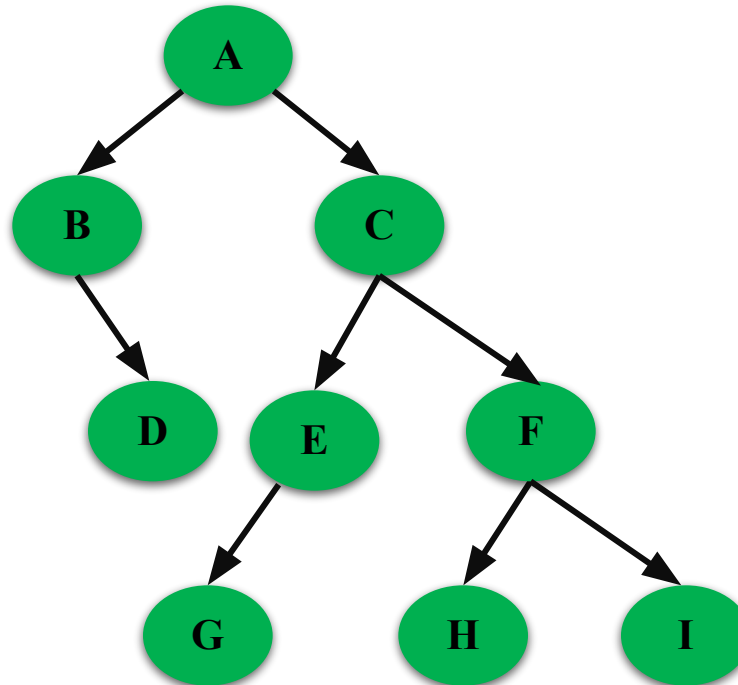
Preorder Traversal

Preorder: Root Left Right
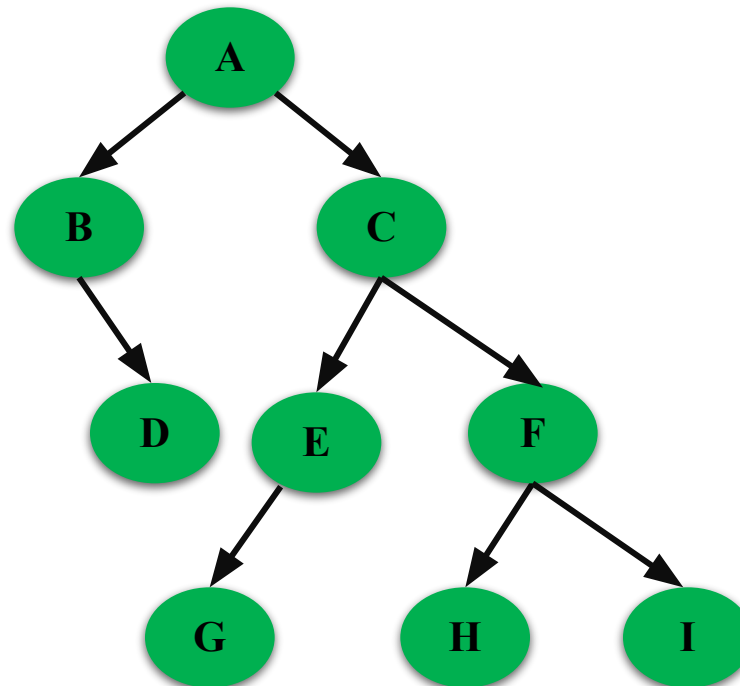
| A | B | D | C | E | G | F | H | I |

Postorder Traversal

Postorder: Left Right Root
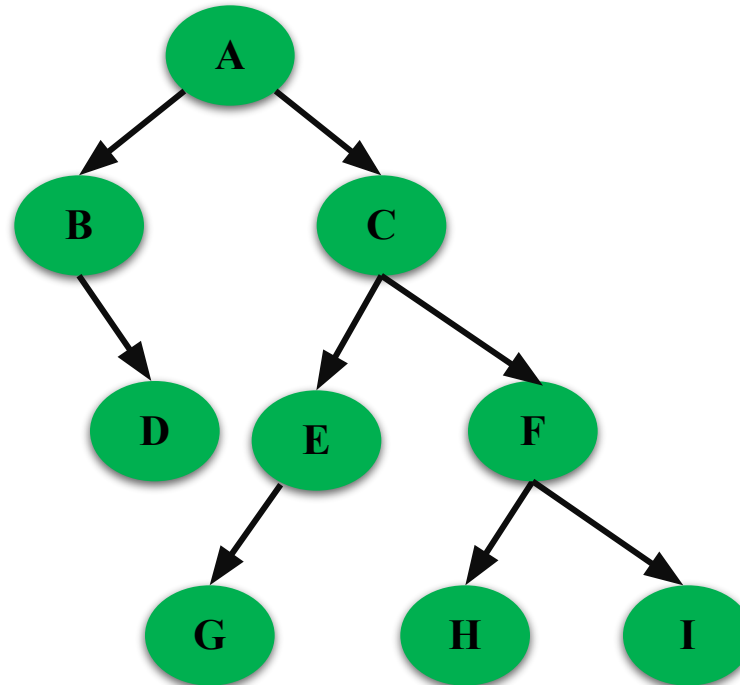
D	B	G	E	H	I	F	C	A

Construct Binary Tree

Preorder: 1 2 4 8 9 10 11 5 3 6 7 (Ro L R)

Inorder: 8 4 10 9 11 2 5 1 6 3 7 (L Ro R)
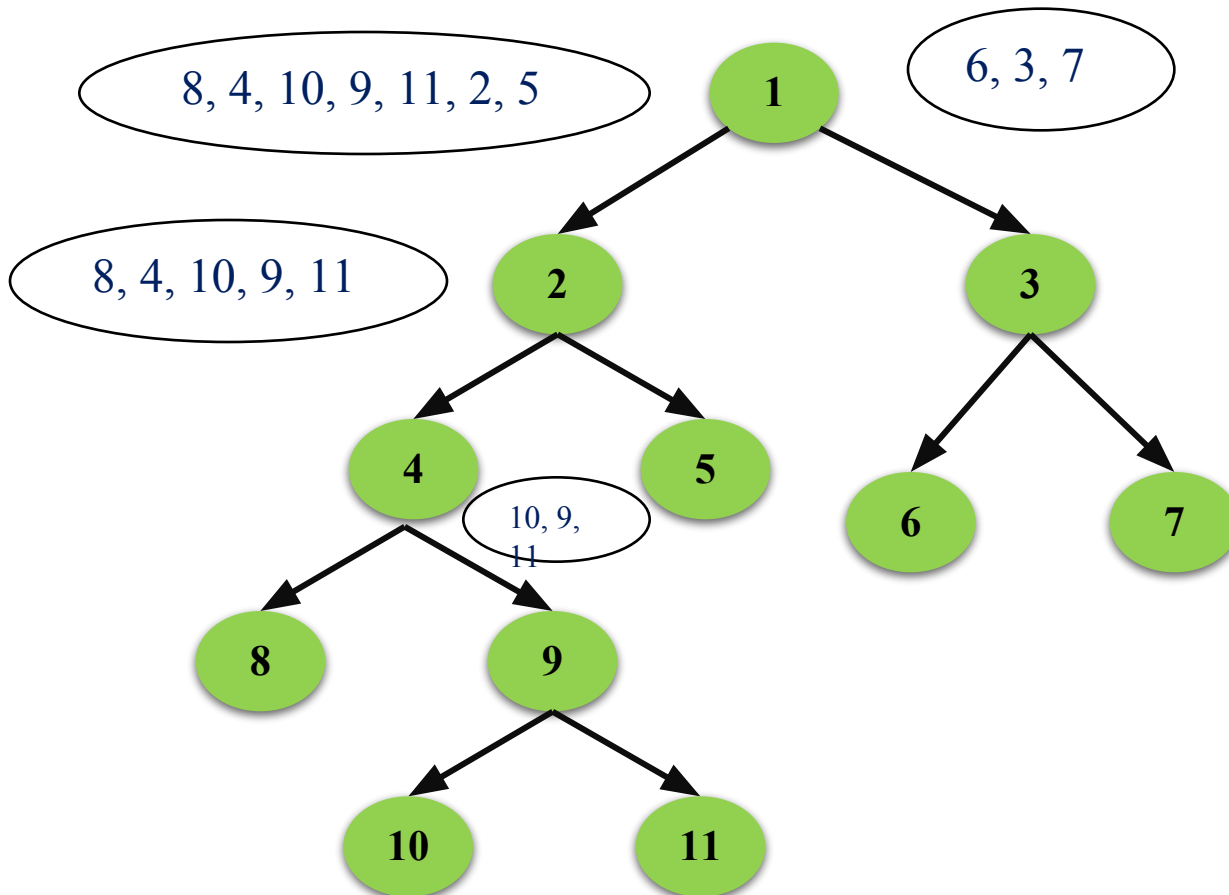
Construct Binary Tree

Postorder: 9 1 2 12 7 5 3 11 4 8 (L R Ro)

Inorder: 9 5 1 7 2 12 8 4 3 11 (L Ro R)
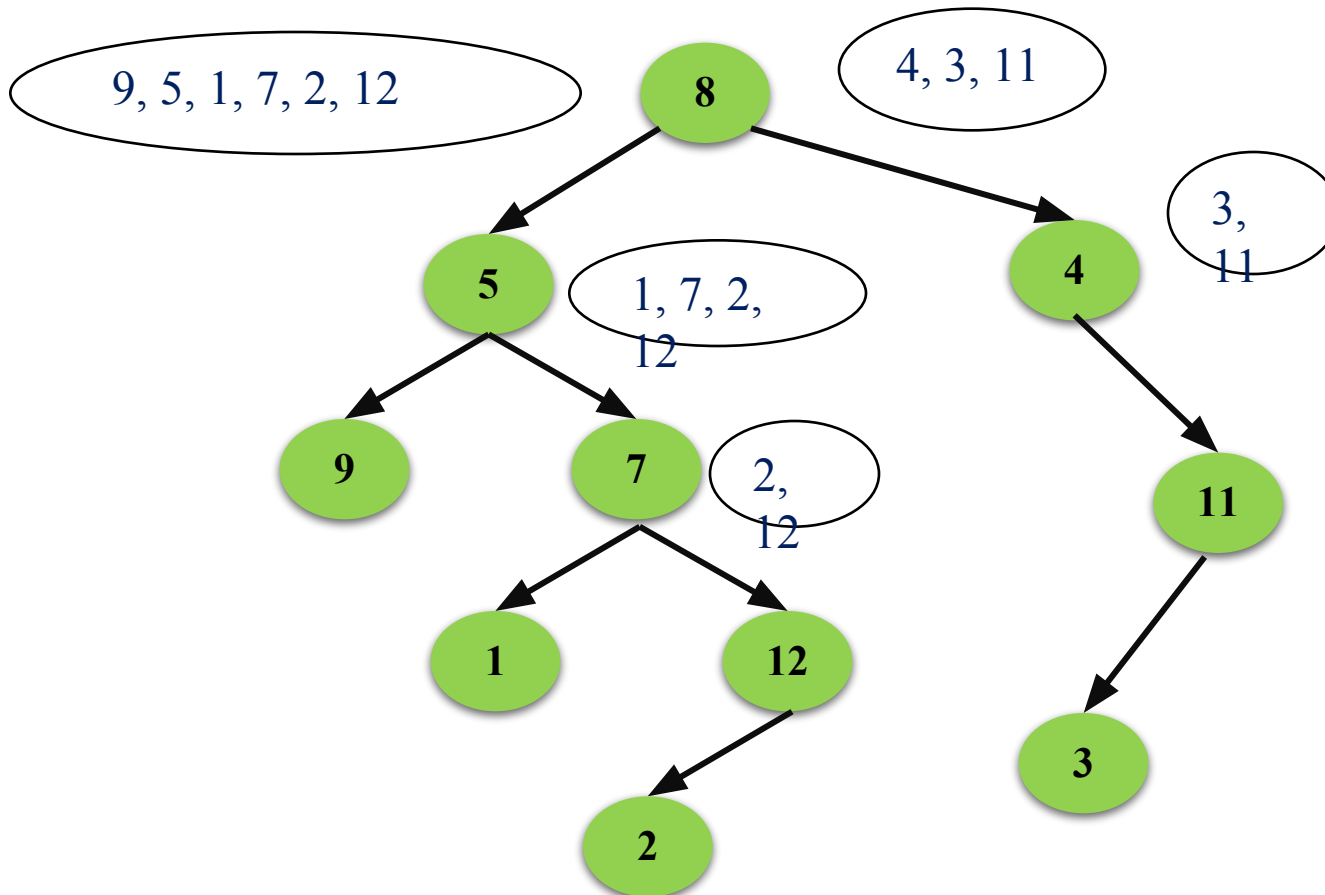
28

Binary Search Tree (BST)

11 6 8 19 4 10 5 17 43 49 31
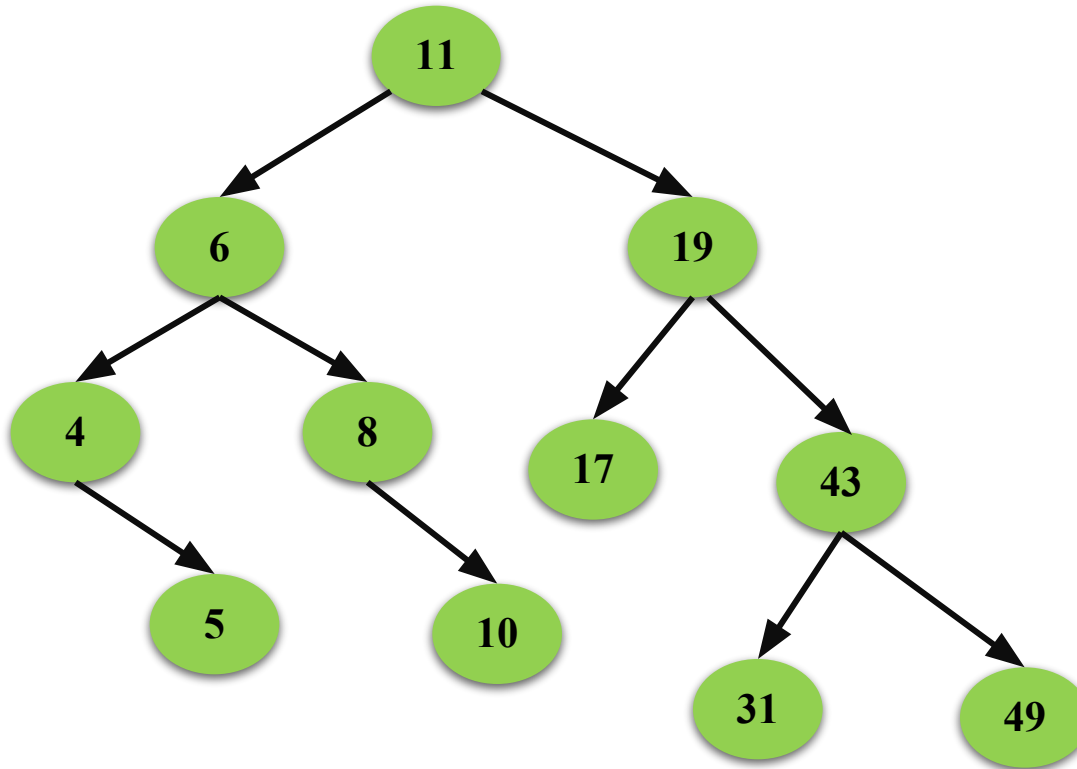
BST Complexity

Insertion
Deletion
Searching

If a Tree have n number of nodes
then there must have $n - 1$ edges

Binary Search Tree (BST)

Deletion: The node you want to delete may have

0 Children   Delete that node   (Delete 5 and 10)

1 Children   Replace the node with its child (Delete 4 or 10)
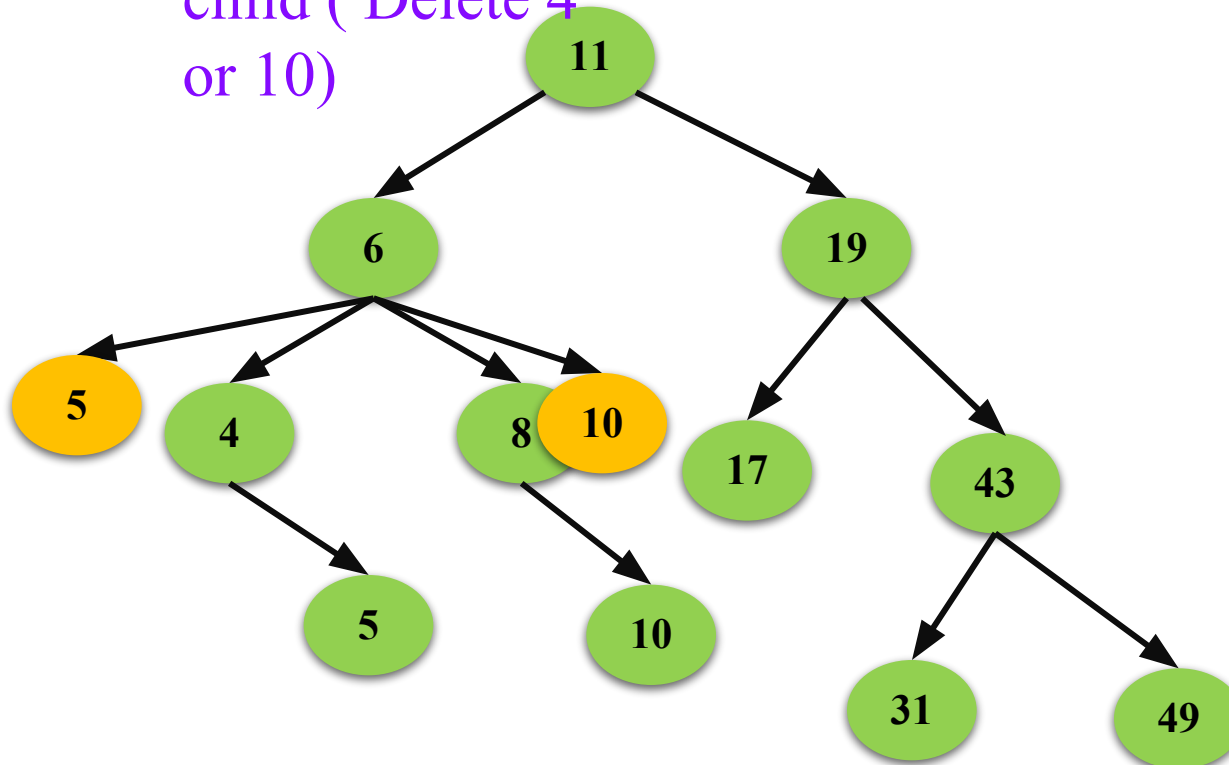
2 Children

Binary Search Tree (BST)

Deletion: The node you want to delete may have

2 Children   ☐ Inorder predecessor   ☐ Inorder successor

Largest element in the Left subtree of the Node being deleted

Smallest element in the Right subtree of the Node being deleted

```
              11
           /      \
          6        19
        /   \     /   \
       4     8   17    43
        \     \       /   \
         5     10    31    49
```
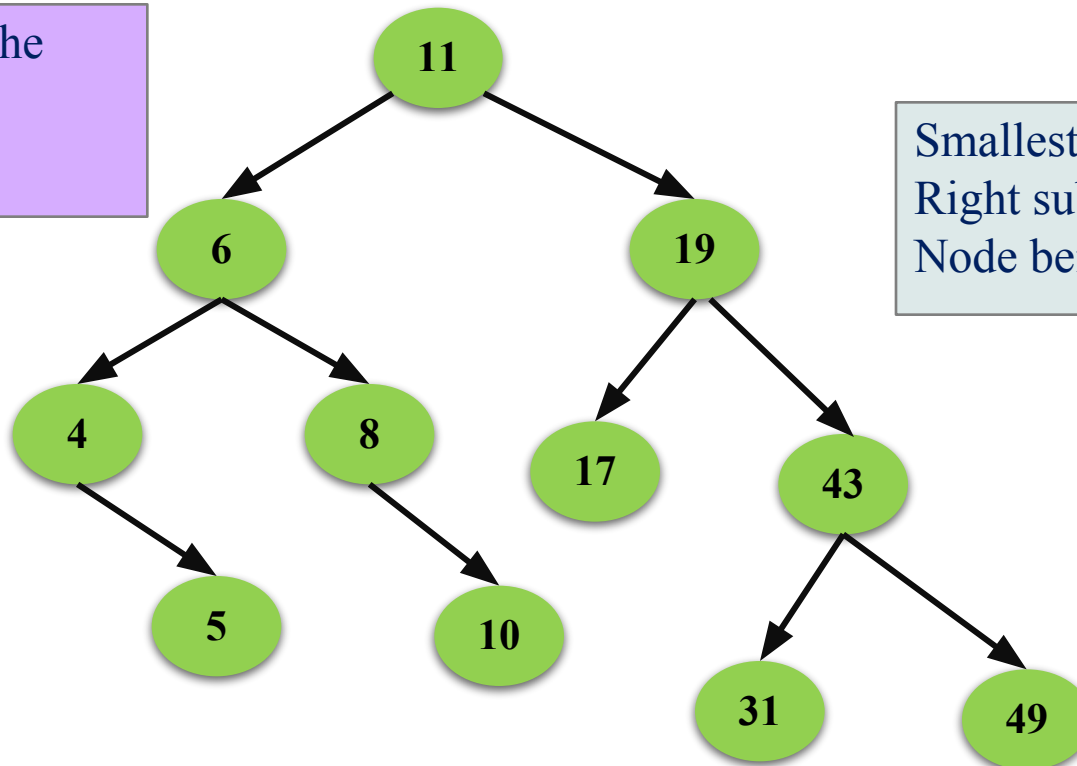
Binary Search Tree (BST)

Suppose, we want to delete node 11

Inorder predecessor
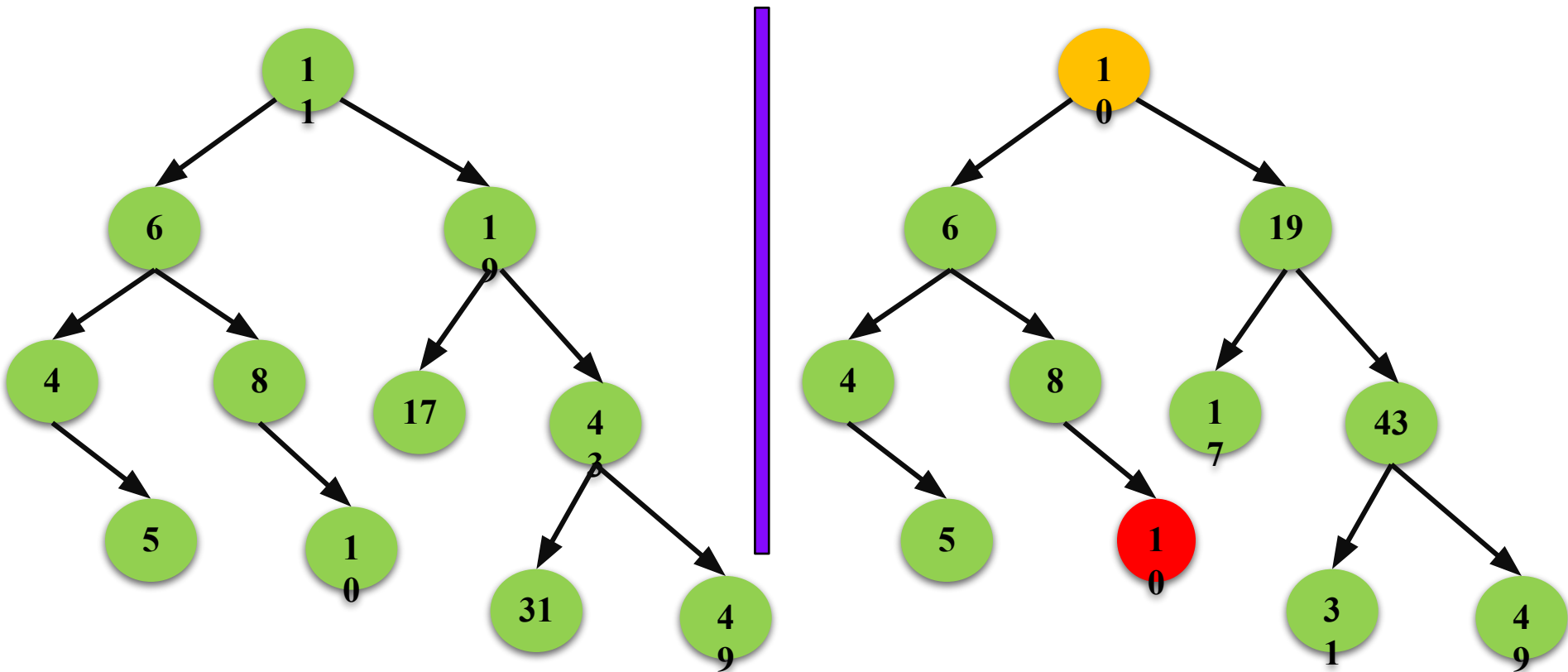
Binary Search Tree (BST)

Suppose, we want to delete node 11

☐ Inorder successor
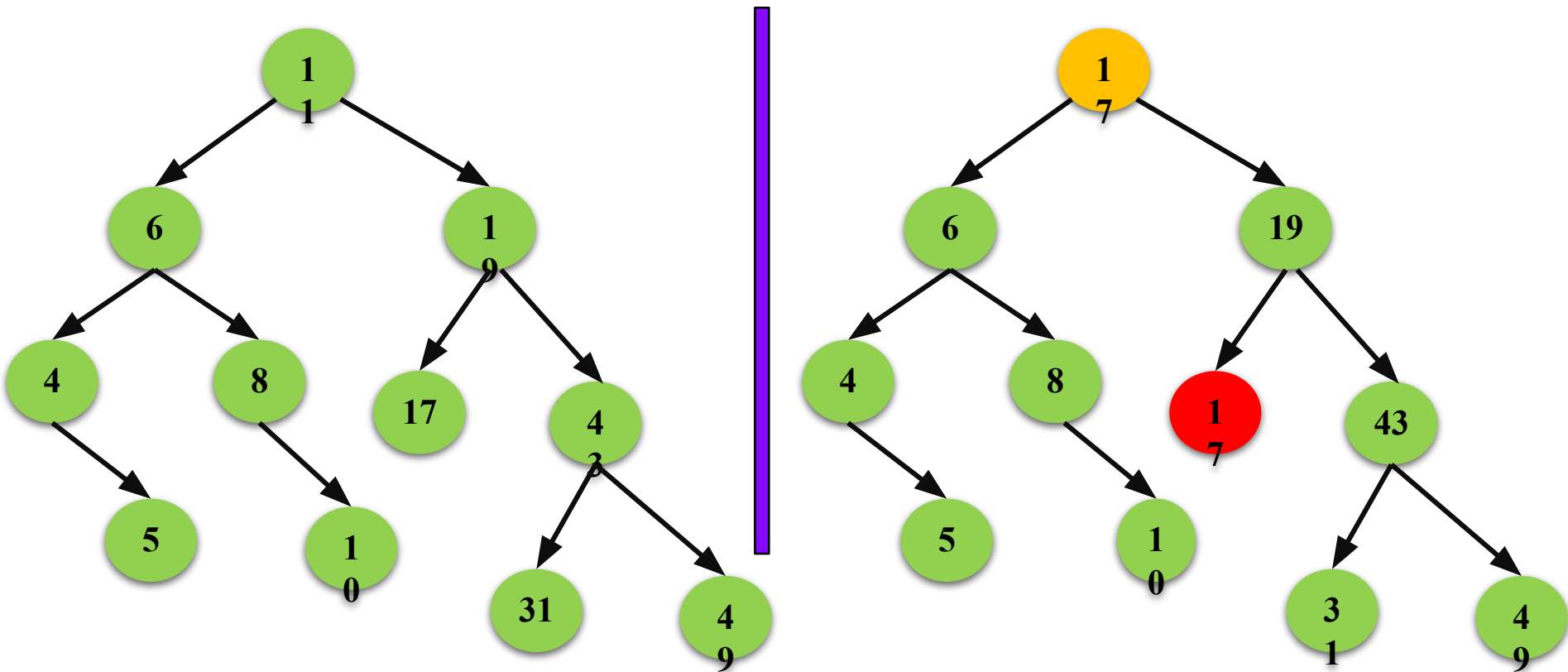
Construct BST

Preorder: 20 16 5 18 17 19 60 85 70 (Ro L R)

Inorder: 5 16 17 18 19 20 60 70 85 (L Ro R)
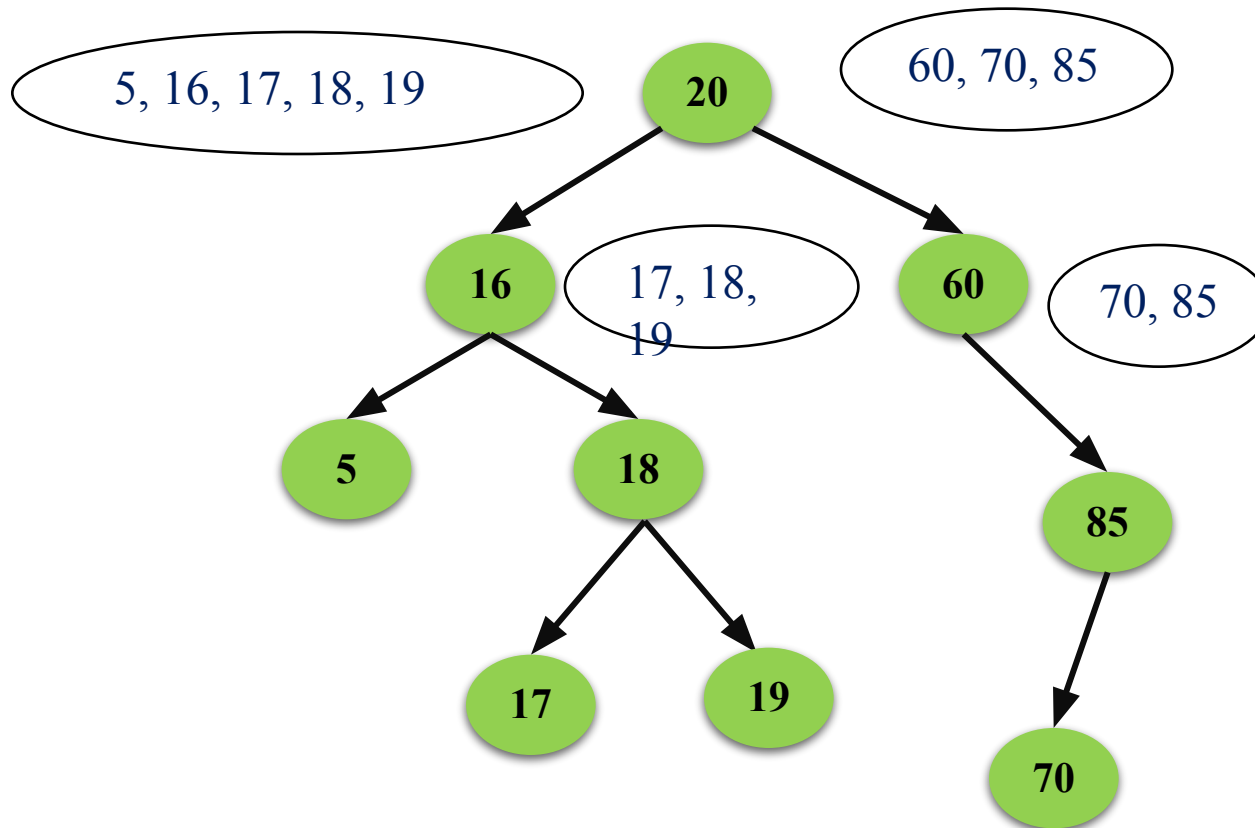
Construct BST

Postorder: 5   17   19   18   16   70   85   60   20   (L R Ro)

Inorder: 5   16   17   18   19   20   60   70   85   (L Ro R)

Self

AVL Tree

If a Tree have n number of nodes then there must have $n - 1$ edges

Balance Factor

AVL tree is a self balancing binary search tree

To balance a binary search tree four situations would arise

AVL Tree

Suppose you want to create a BST using ⬜ x, y, z

Left Rotation

AVL Tree

Suppose you want to create a BST using ▢ z, y, x

Right Rotation

AVL Tree

Suppose you want to create a BST using ▯ x, z, y

Right Rotation

Left Rotation

x -2

z 0

y 0

x -2

y -1

z 0

y 0

x 0

z 0

AVL Tree

Suppose you want to create a BST using □ z, x, y

z   2

x   -0

y   0

Left Rotation

z   2

y   0

x   0

Right Rotation

y   0

x   0

z   0

41

Construct AVL Tree

14   17   11   7   53   4   13   12   8   60   19   16   20

Construct AVL Tree

14   17   11   7   53   4   13   12   8   60   19   16   20

Construct AVL Tree

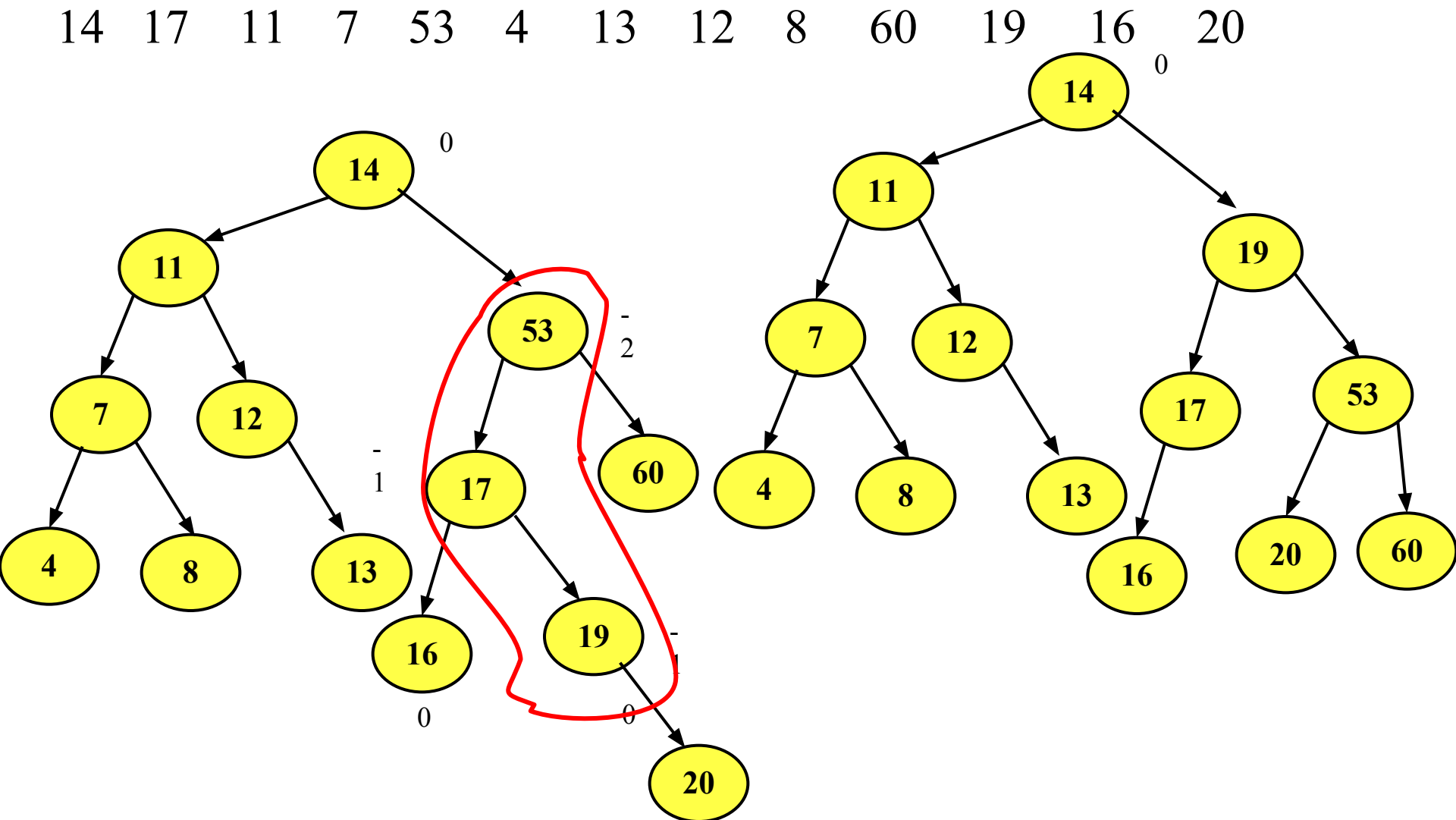14   17   11   7   53   4   13   12   8   60   19   16   20

Construct AVL Tree
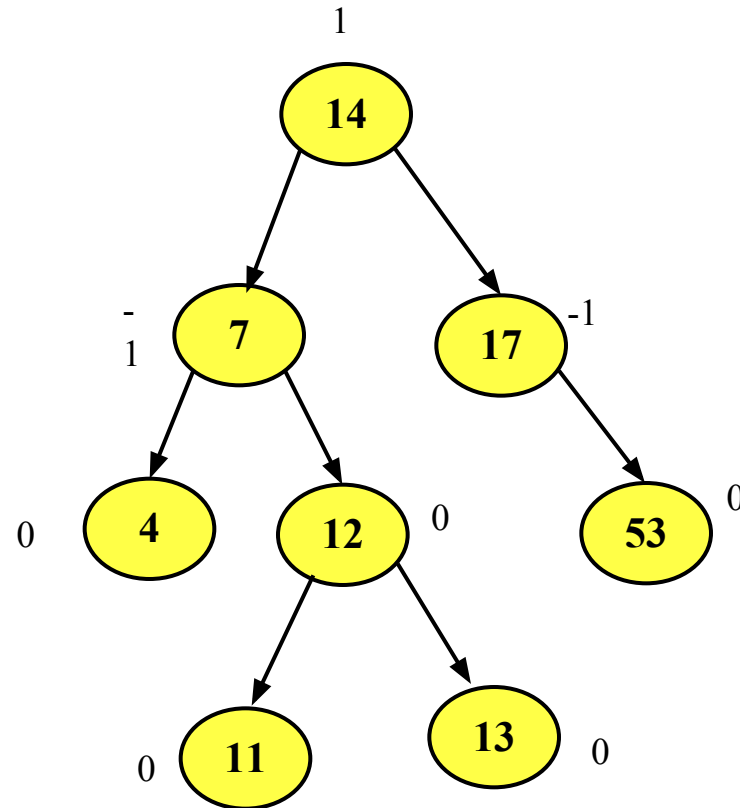
14 17 11 7 53 4 13 12 8 60 19 16 20

Construct AVL Tree
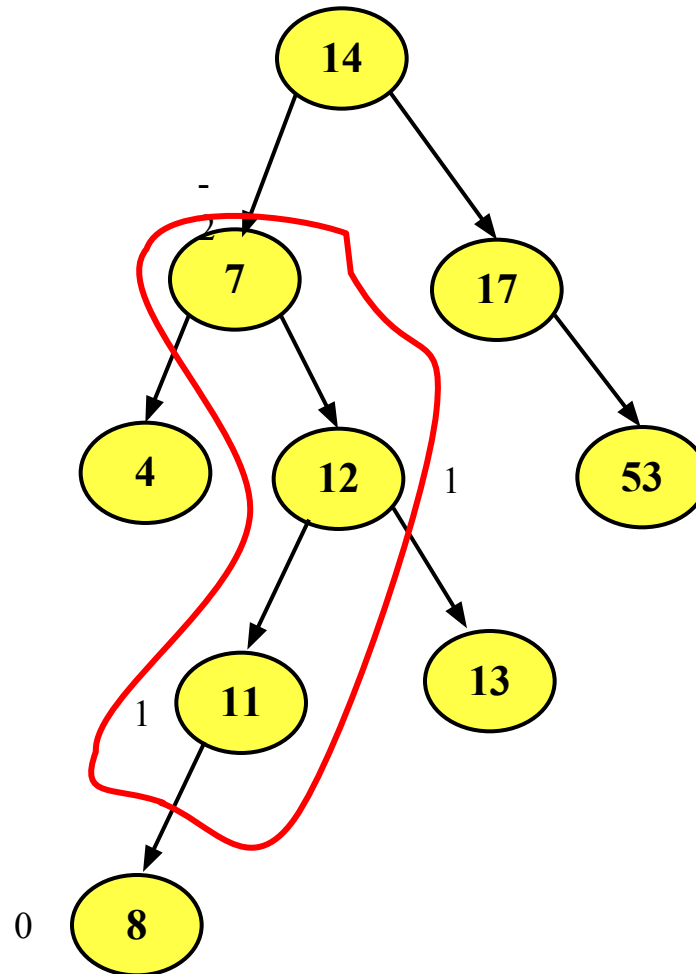
14   17   11   7   53   4   13   12   8   60   19   16   20

14   17   11   7   53   4   13   12   8   60   19   16   20
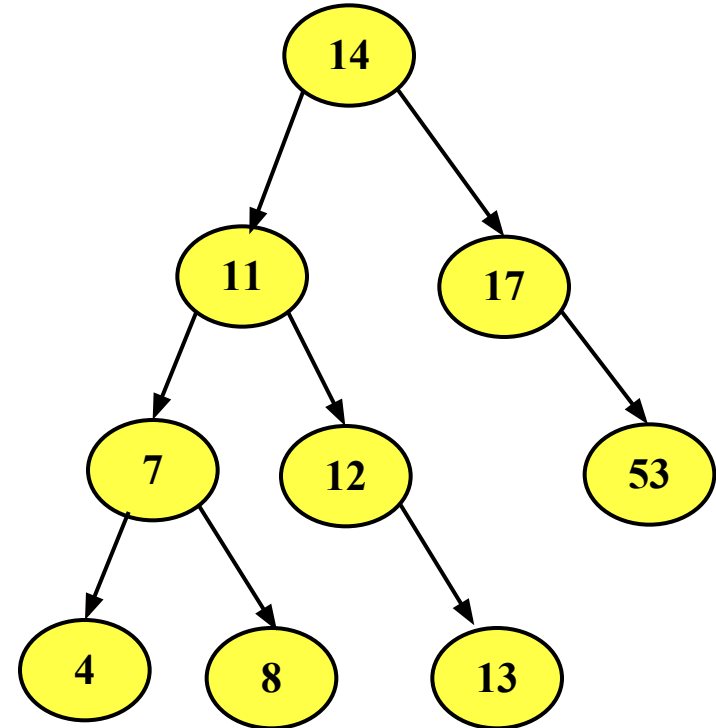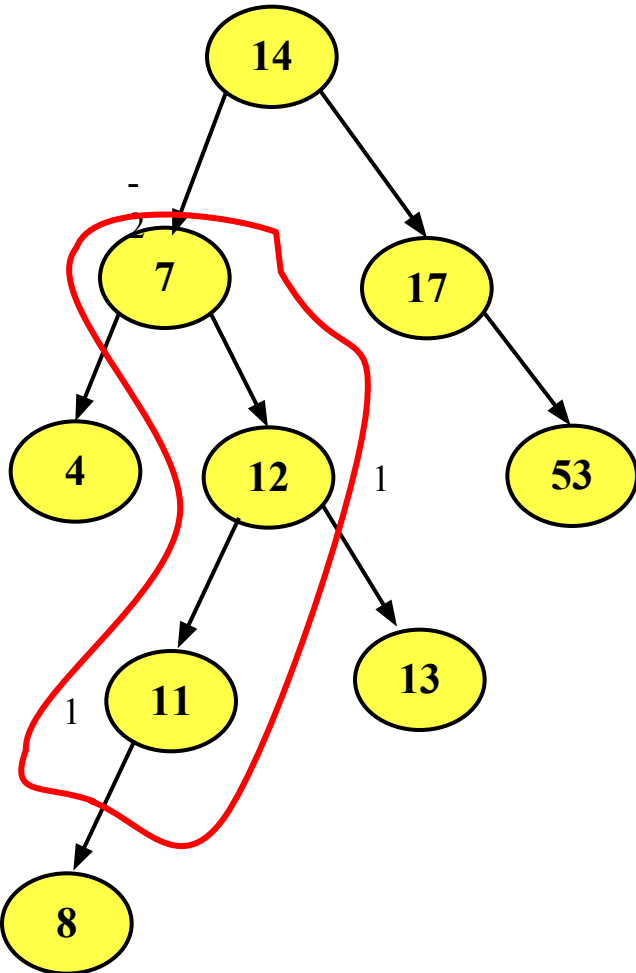
Construct AVL Tree

14   17   11   7   53   4   13   12   8   60   19   16   20

Construct AVL Tree

14   17   11   7   53   4   13   12   8   60   19   16   20

Construct AVL Tree

14   17   11   7   53   4   13   12   8   60   19   16   20

Construct AVL Tree

14   17   11   7   53   4   13   12   8   60   19   16   20

Construct AVL Tree

14 17 11 7 53 4 13 12 8 60 19 16 20

Deletion in AVL Tree

Deletion is same as binary search tree

After every deletion you have to balance the tree

8 7 11 14 17

Delete 8
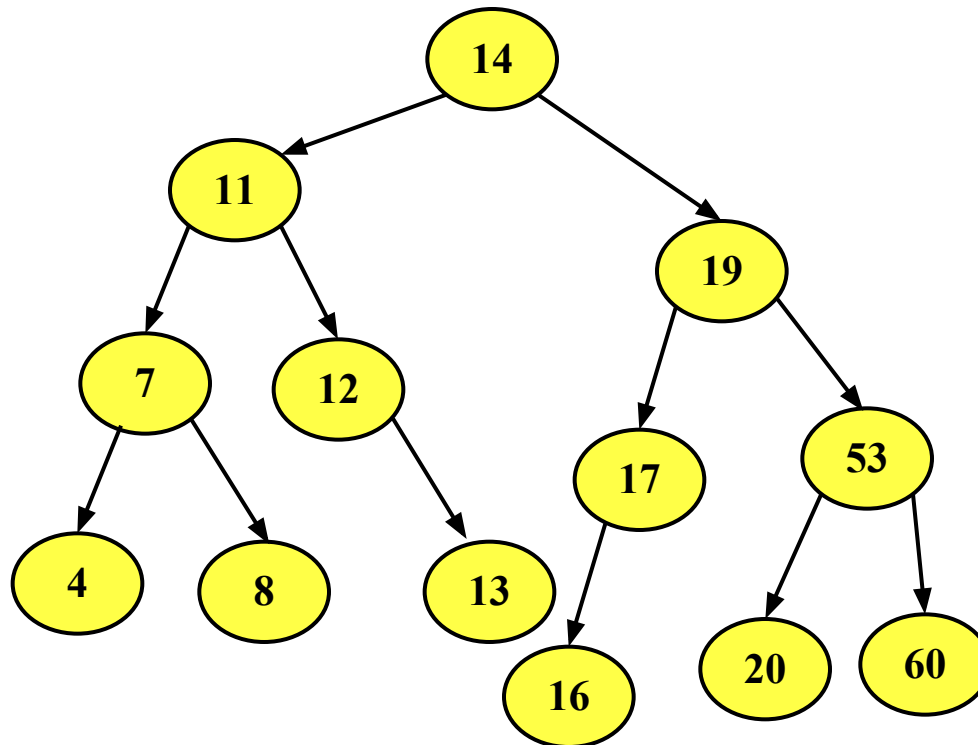
Deletion in AVL Tree

8 7 11 14 17

Delete 7
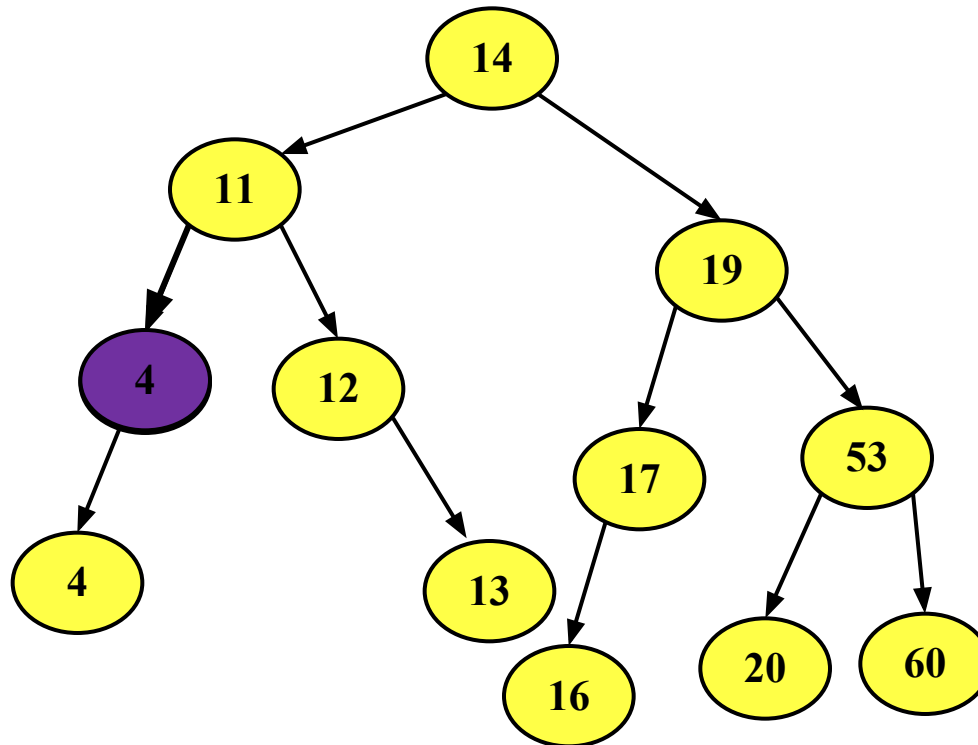
8   7   11   14   17

Delete 11

Deletion in AVL Tree

8   7   11   14   17

Delete 11

$\frac{-}{2}$

Replace by inorder predecessor or successor

8   7   11   14   17

Delete 11

14

$\frac{-}{2}$ 4

19

12

17

53

13

16

20   60
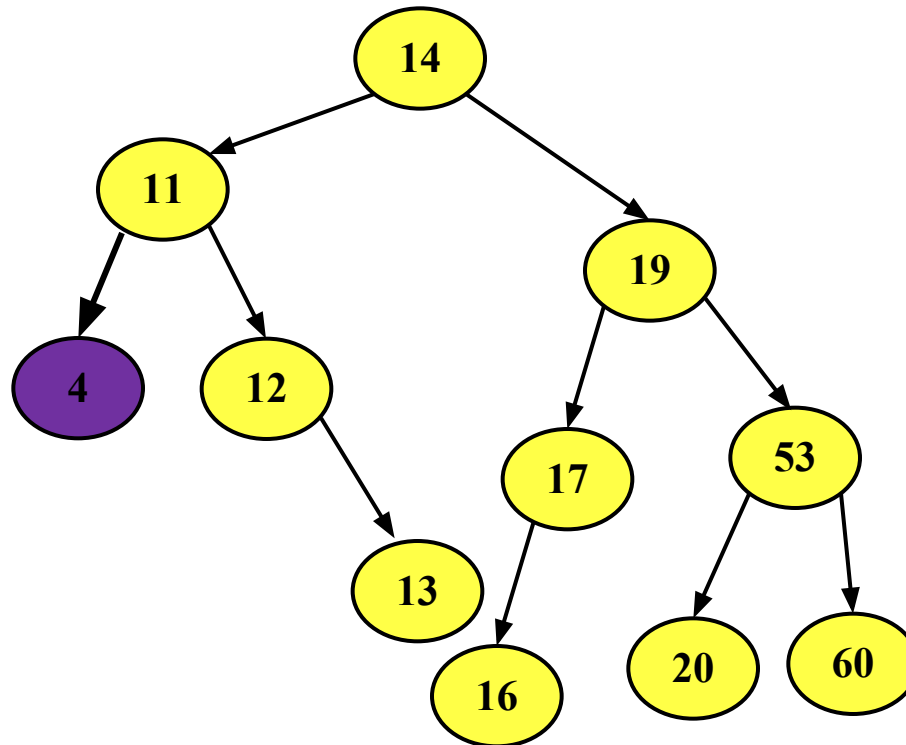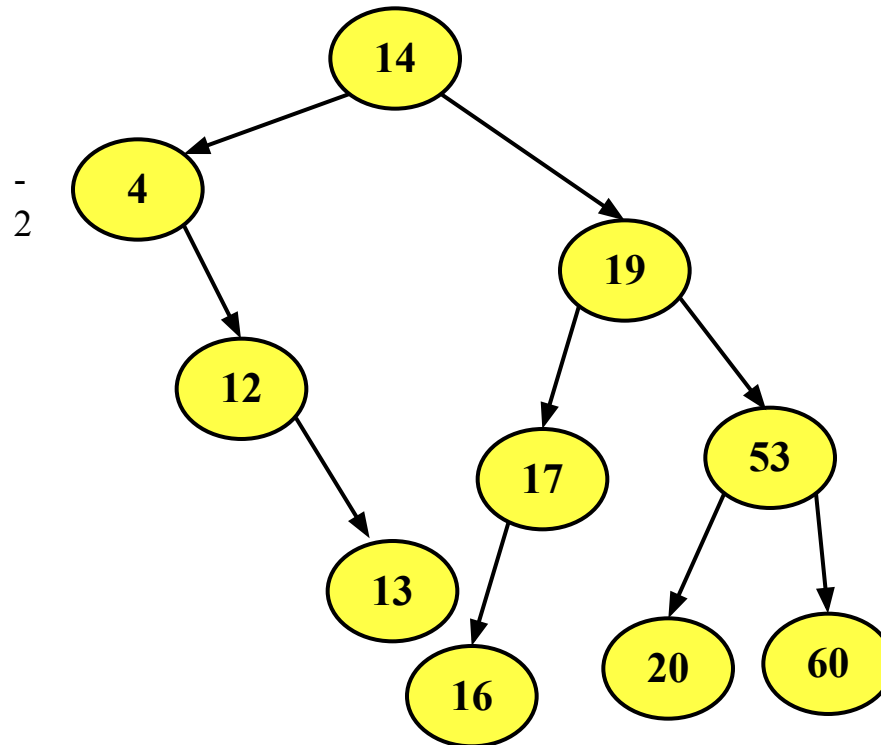
Replace by inorder predecessor or successor

Deletion in AVL Tree

8 7 11 14 17

Balanced
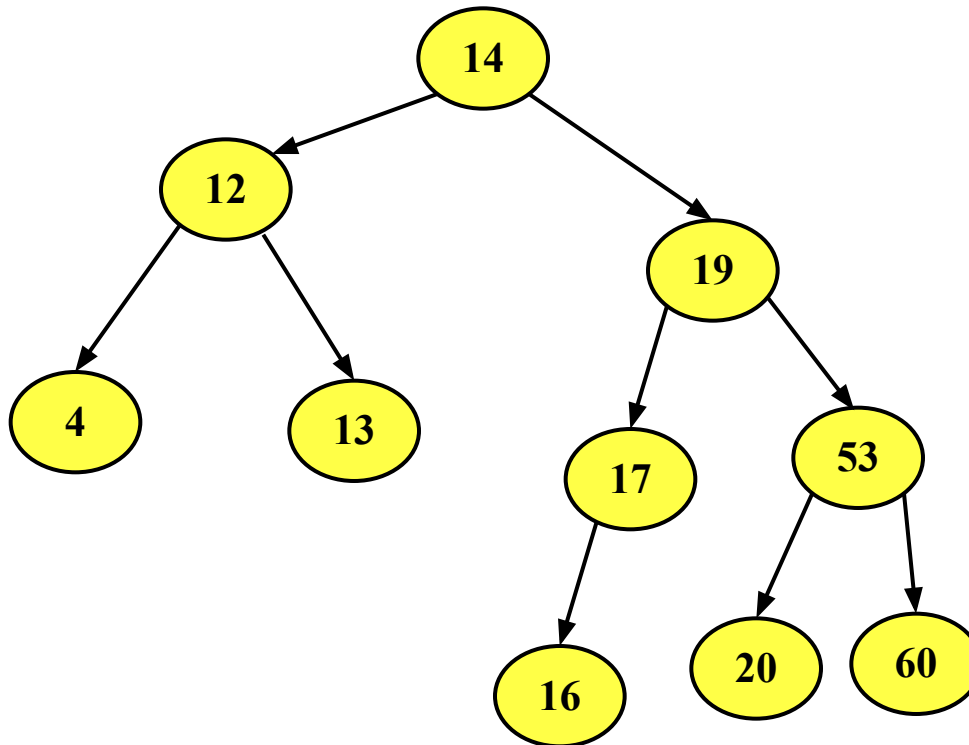
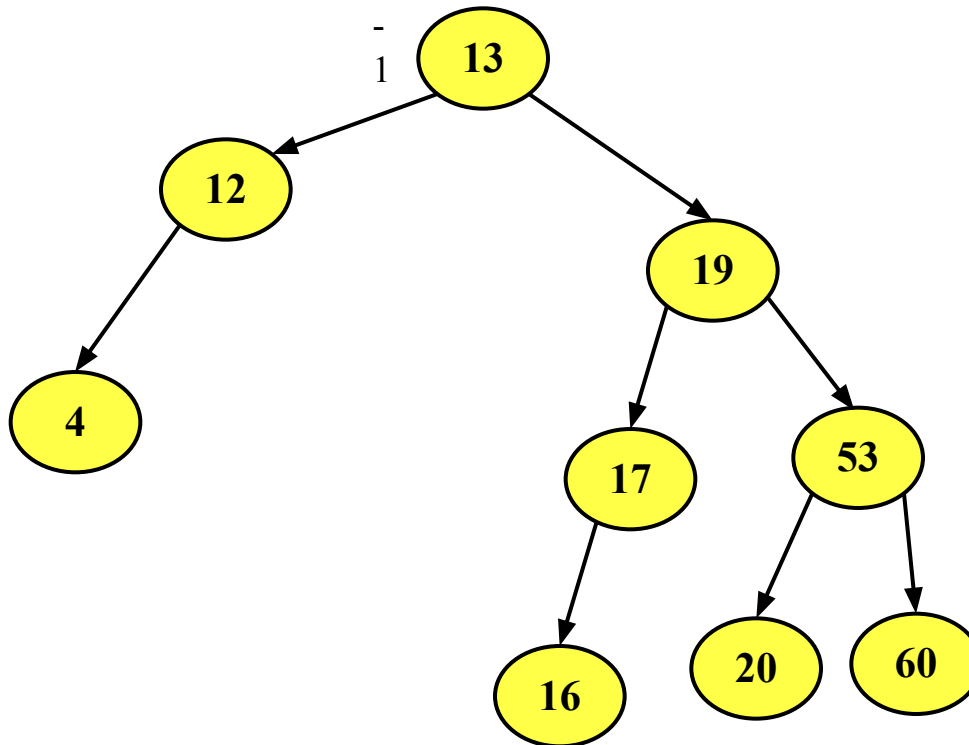Performed Left Rotation

Deletion in AVL Tree
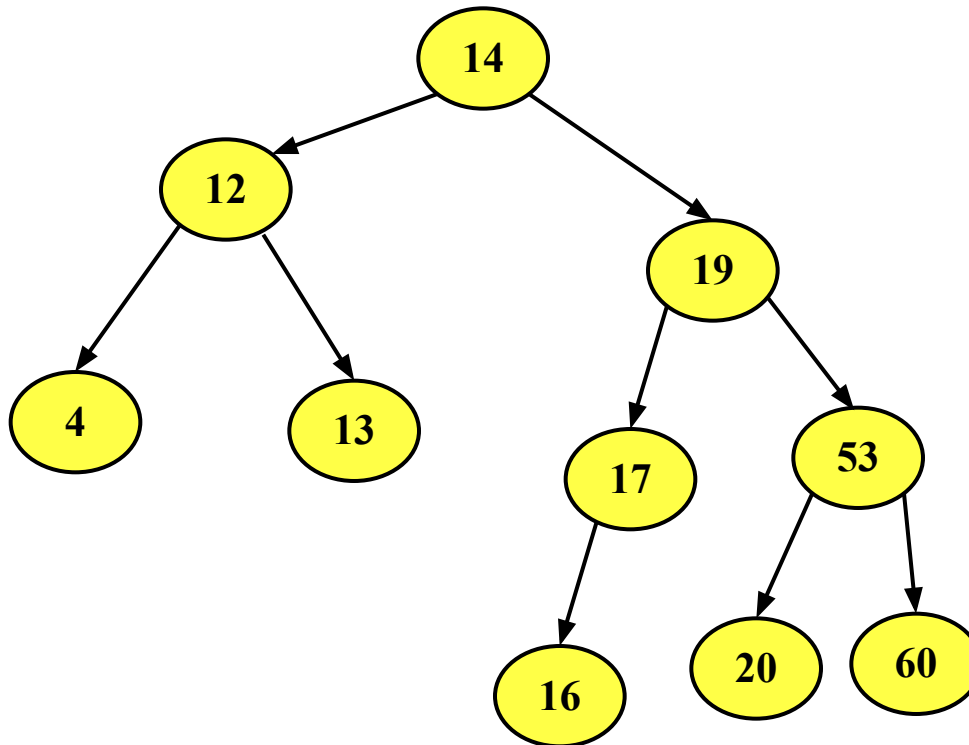
8   7   11   14   17

Delete 14

Replace by inorder predecessor or successor

8   7   11   14   17

Delete 14

```
          ‾
          1   13
             /  \
           12    19
          /     /  \
         4    17    53
             /      / \
           16     20   60
```

Replace by inorder predecessor or successor

8 7   11   14   17

Delete 17
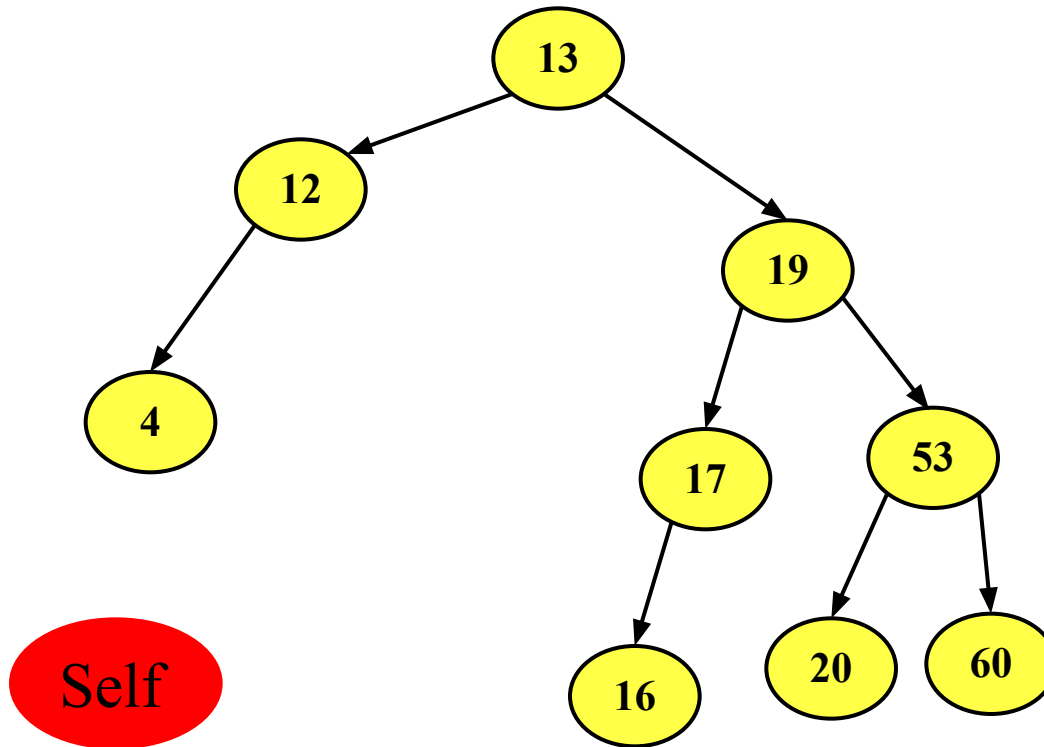
AVL Tree Complexity

Insertion
Deletion
Searching

If a Tree have n number of nodes then there must have $n - 1$ edges

Search ⬜ Best Case ⬜ O(1)

B-Tree

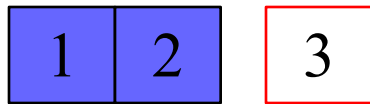If a Tree have n number of nodes then there must have $n - 1$ edges

B-Tree

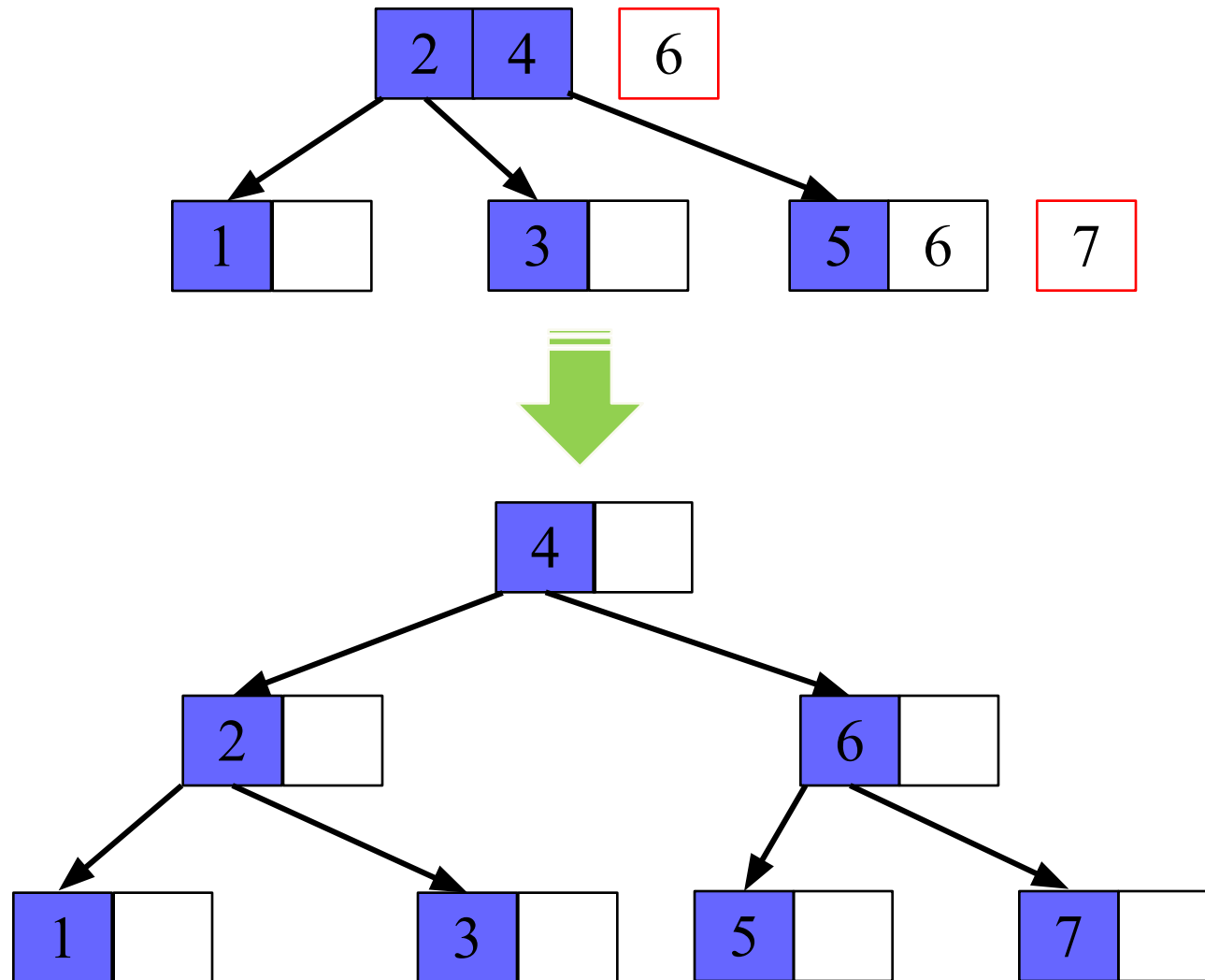If a Tree have n number of nodes then there must have $n - 1$ edges

Insertion in B-Tree

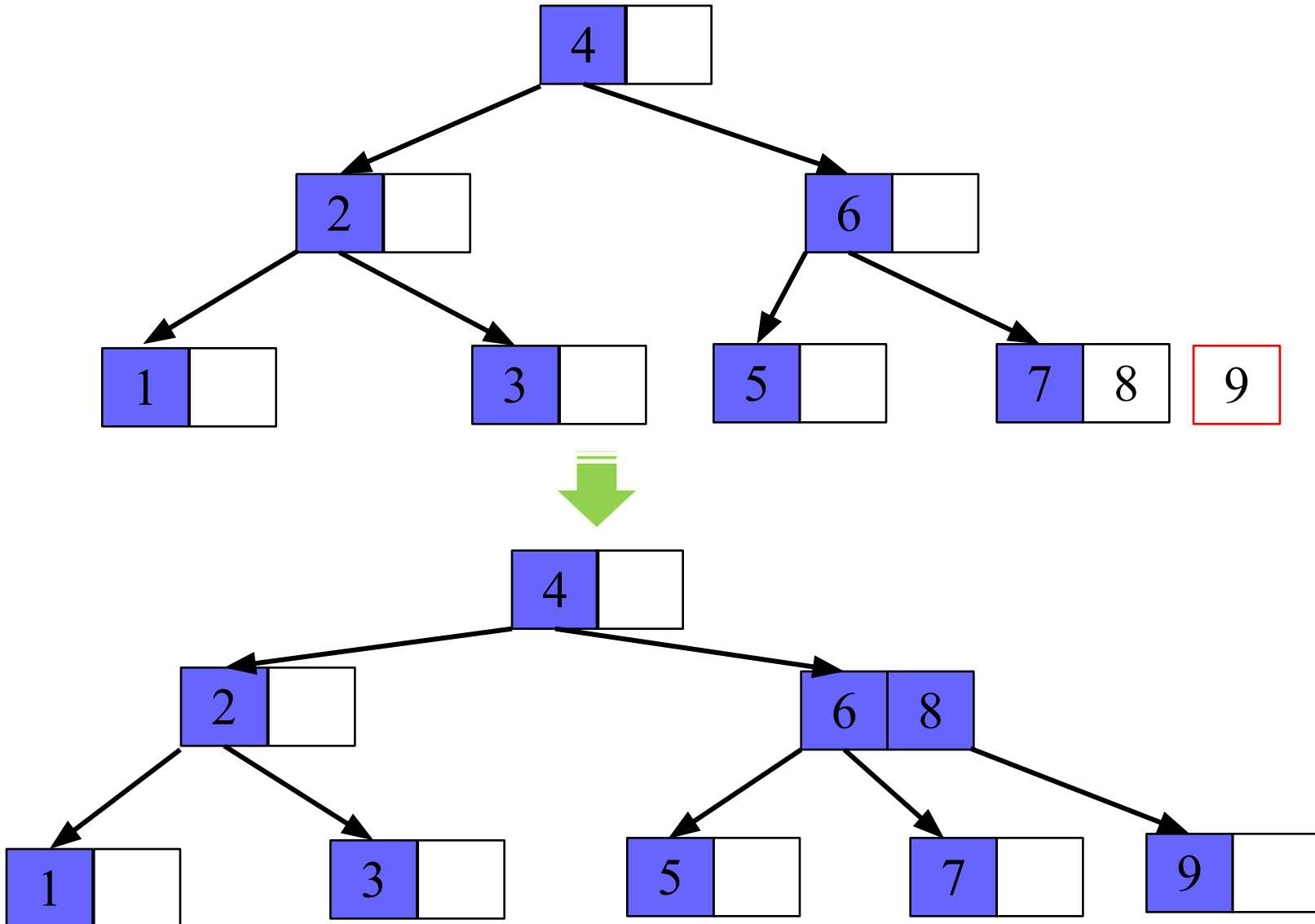Create a B – tree of order 3 by inserting values from 1 to 10

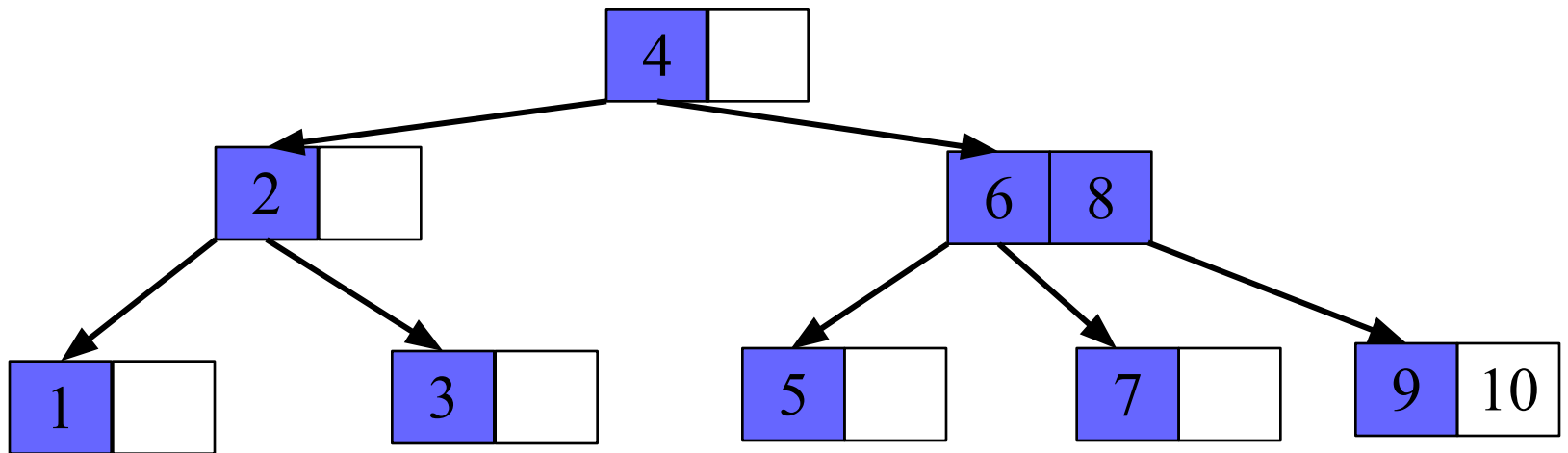If a Tree have n number of nodes then there must have $n - 1$ edges
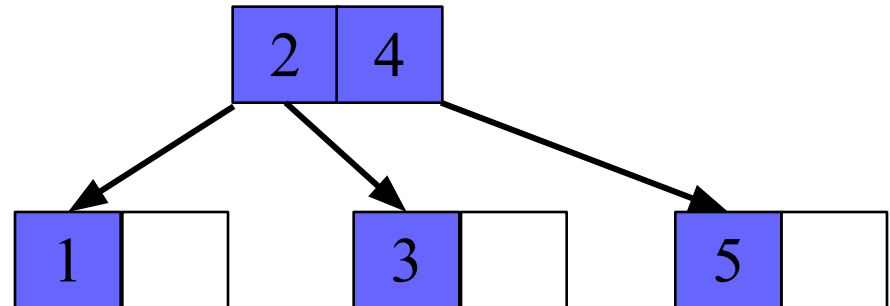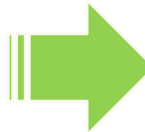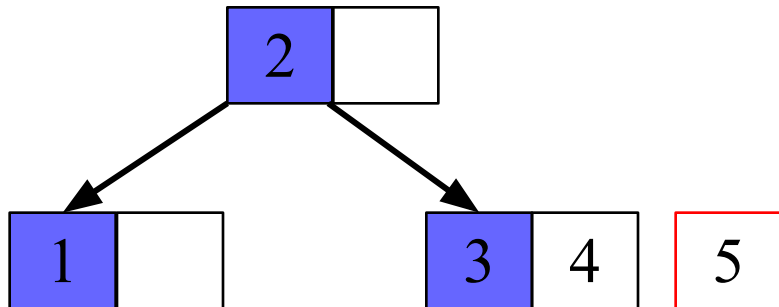
Insertion in B-Tree
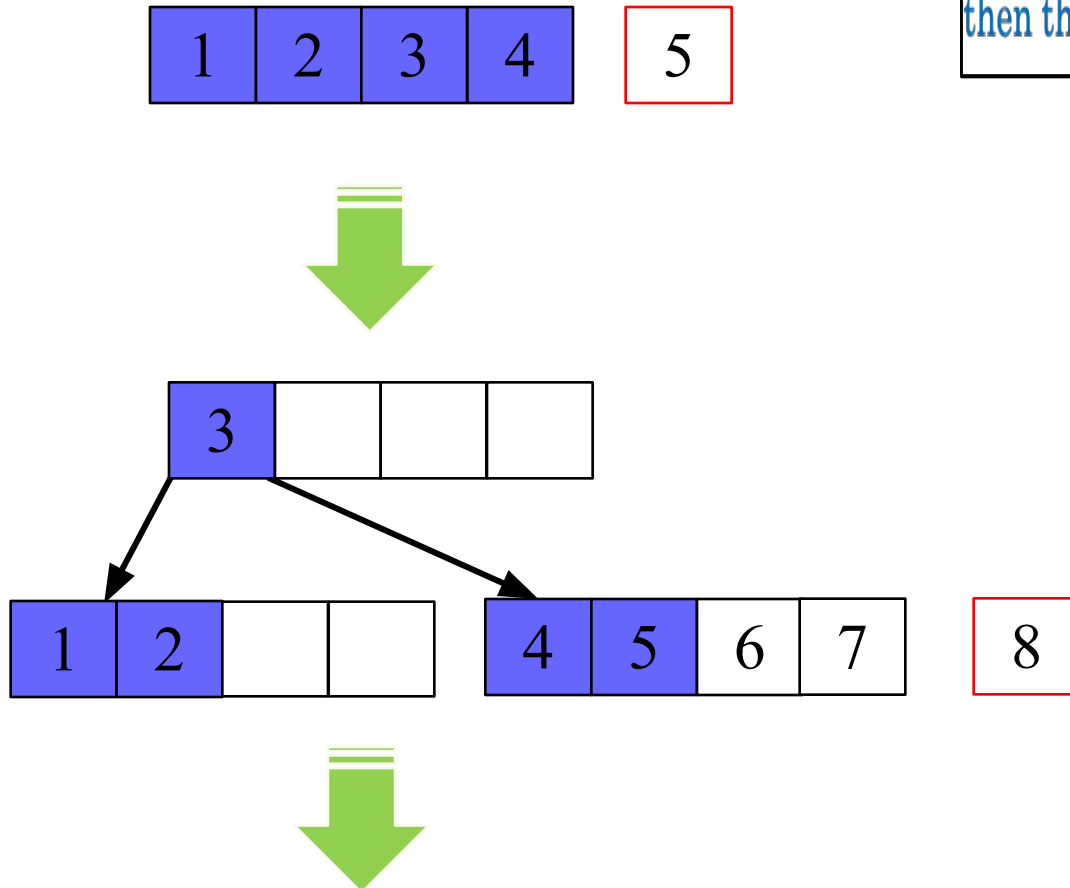
Insertion in B-Tree
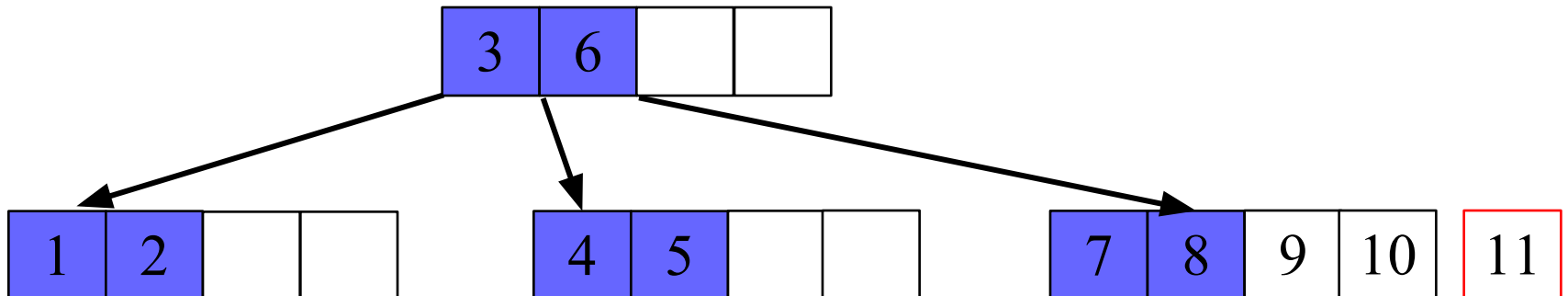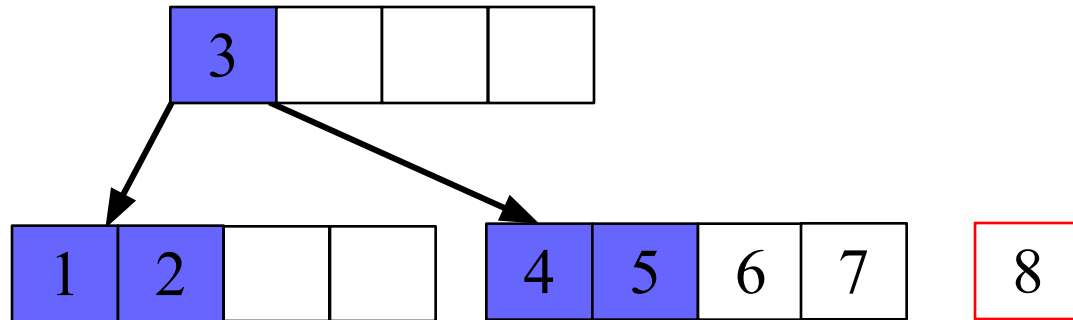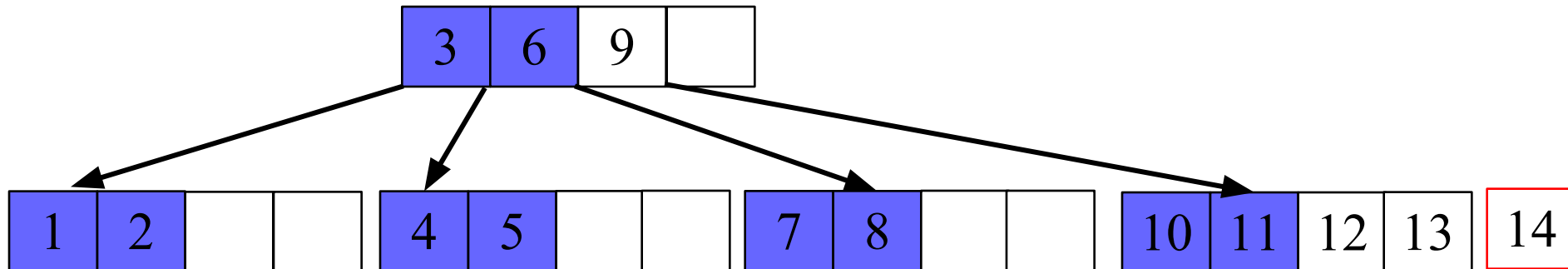
Insertion in B-Tree

Insertion in B-Tree

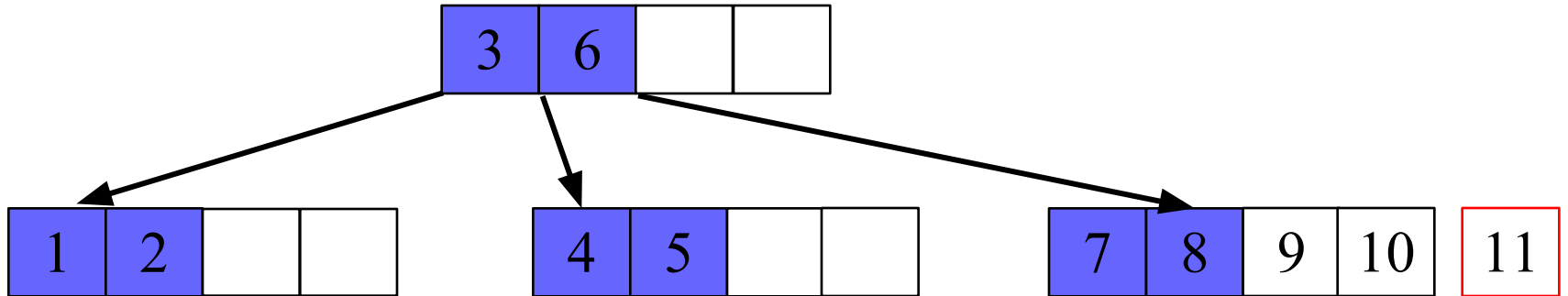Create a B – tree of order 5 by inserting values from 1 to 20

If a Tree have n number of nodes then there must have $n - 1$ edges
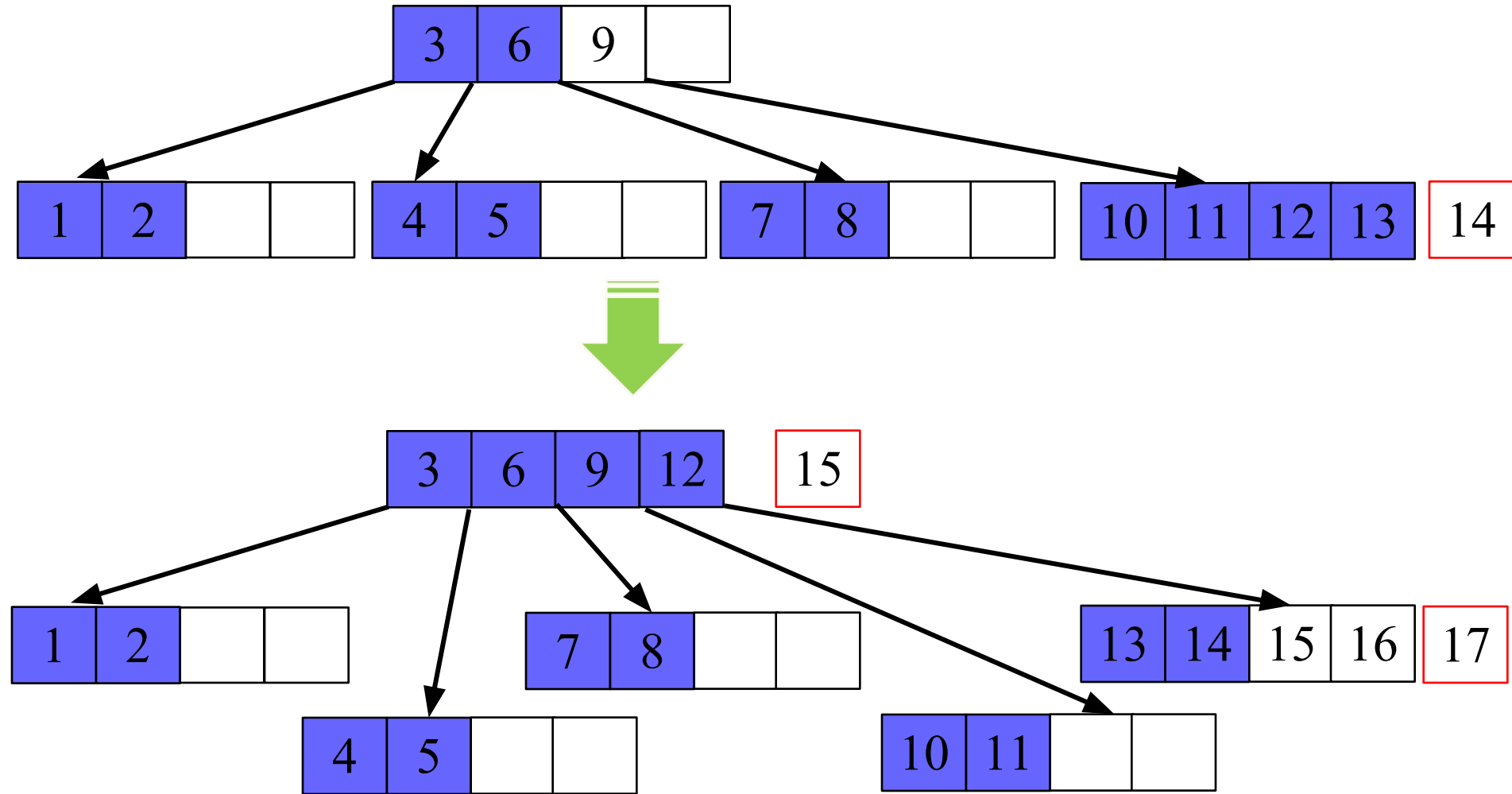
| 1 | 2 | 3 | 4 |

| 5 |

| 3 | | | |

| 1 | 2 | | |

| 4 | 5 | 6 | 7 |

| 8 |

69

Insertion in B-Tree

Insertion in B-Tree

Insertion in B-Tree
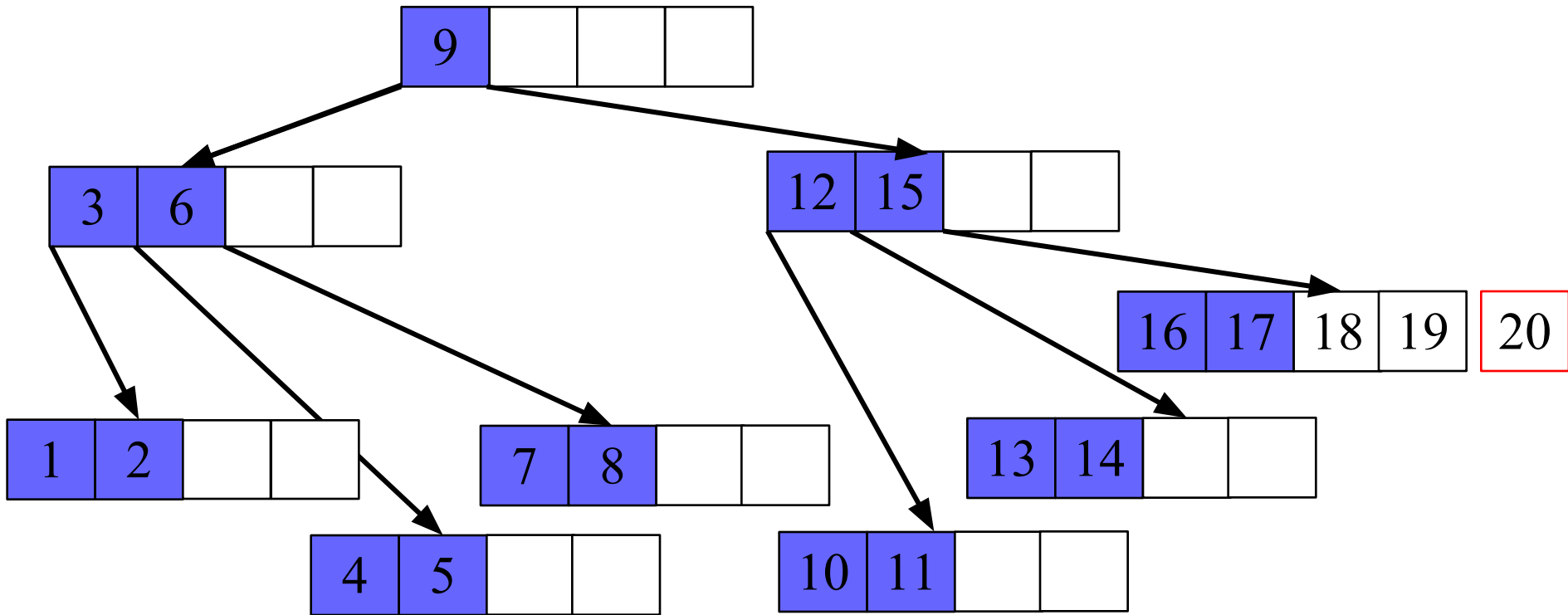
Insertion in B-Tree

Last Split (Self)

Insertion in B-Tree

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

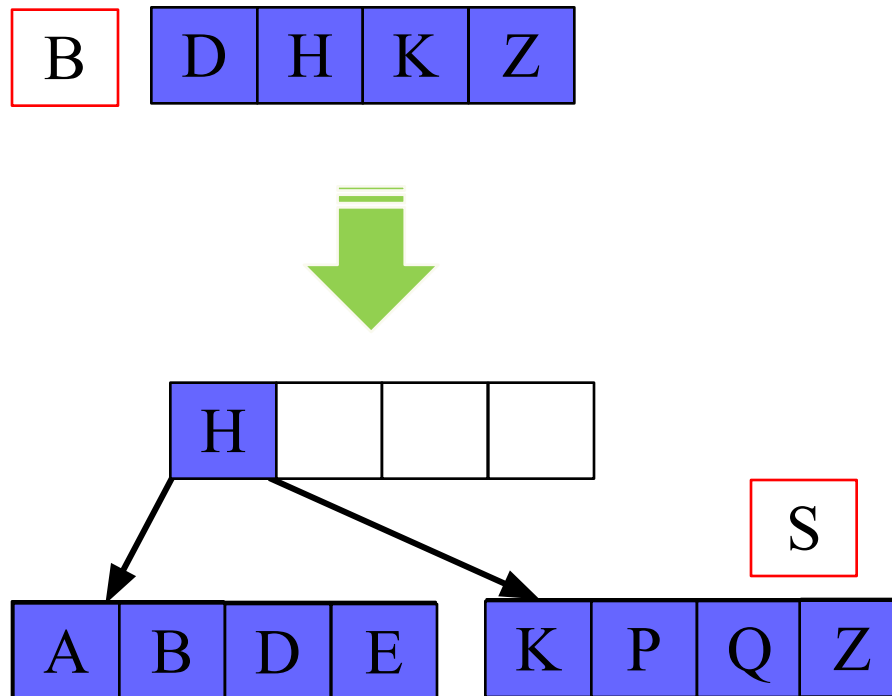| B | | D | H | K | Z |

| H | | | |

| S |

| A | B | D | E | | K | P | Q | Z |

Insertion in B-Tree

Create a B – tree of order 5 with the following set of data
D, H, Z, K, B, P, Q, E, A, S, W, T, C, L, N, Y, M

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

| H | | | |

S

| A | B | D | E |

| K | P | Q | Z |

| H | Q | | |

C

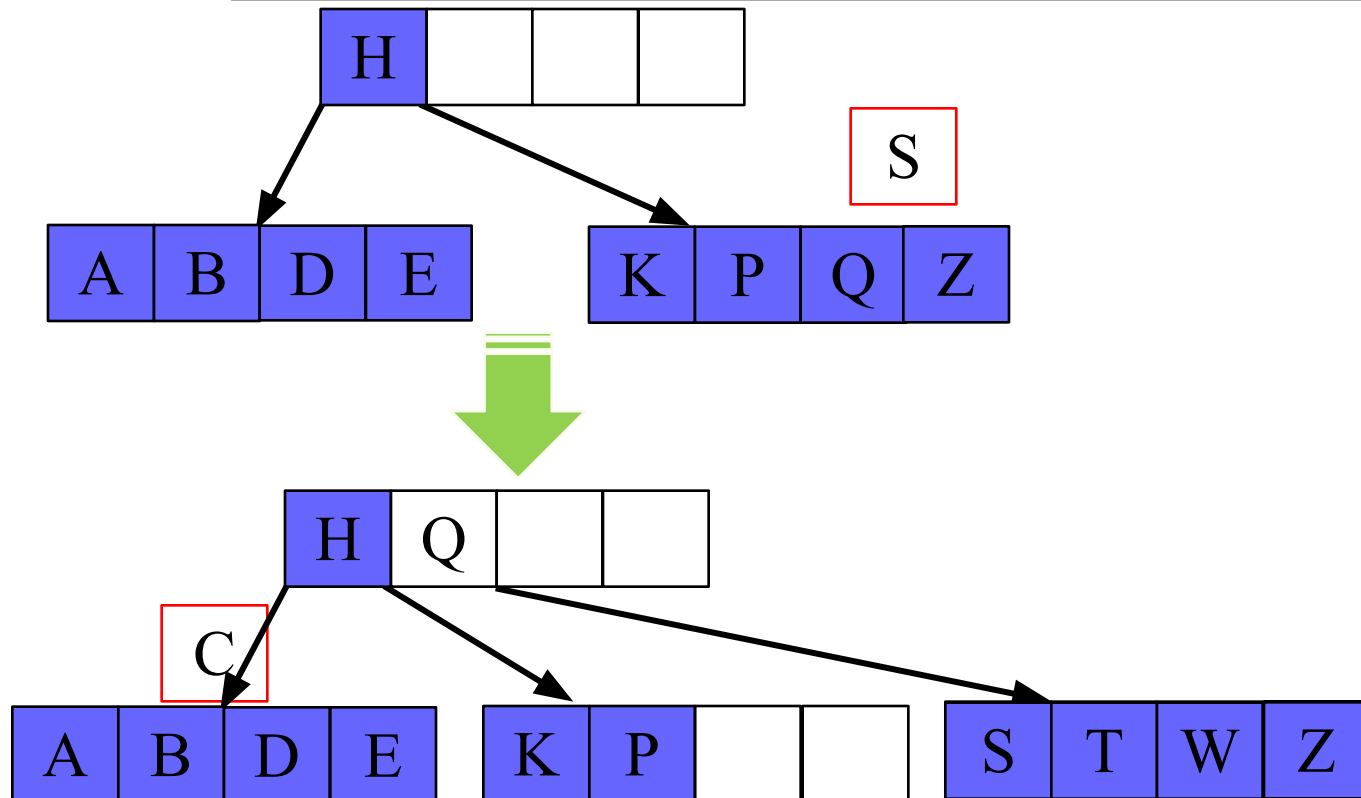| A | B | D | E |

| K | P | | |

| S | T | W | Z |

Insertion in B-Tree

Create a B – tree of order 5 with the following set of data
D, H, Z, K, B, P, Q, E, A, S, W, T, C, L, N, Y, M

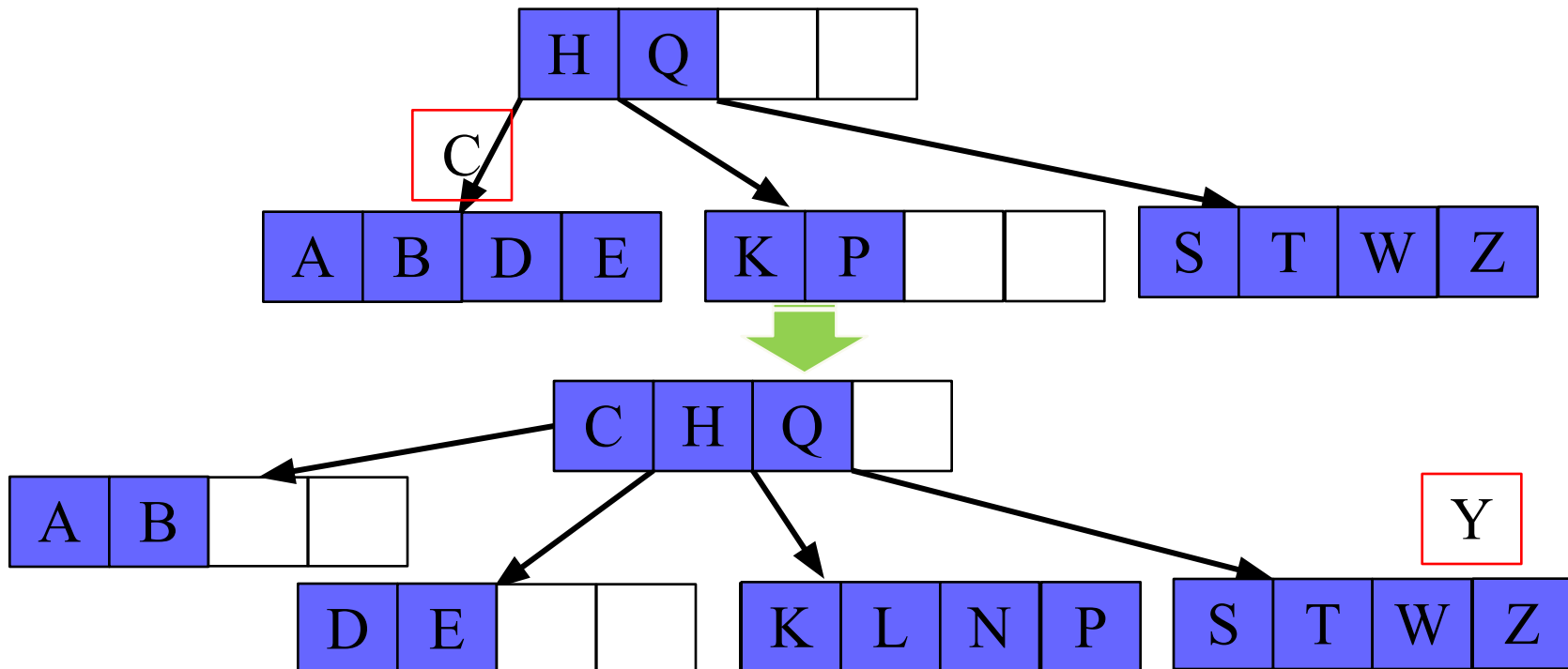A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

Insertion in B-Tree

Create a B – tree of order 5 with the following set of data
    D, H, Z, K, B, P, Q, E, A, S, W, T, C, L, N, Y, M

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

M

| C | H | Q | W |

| A | B | | |

| D | E | | |

M

| K | L | N | P |

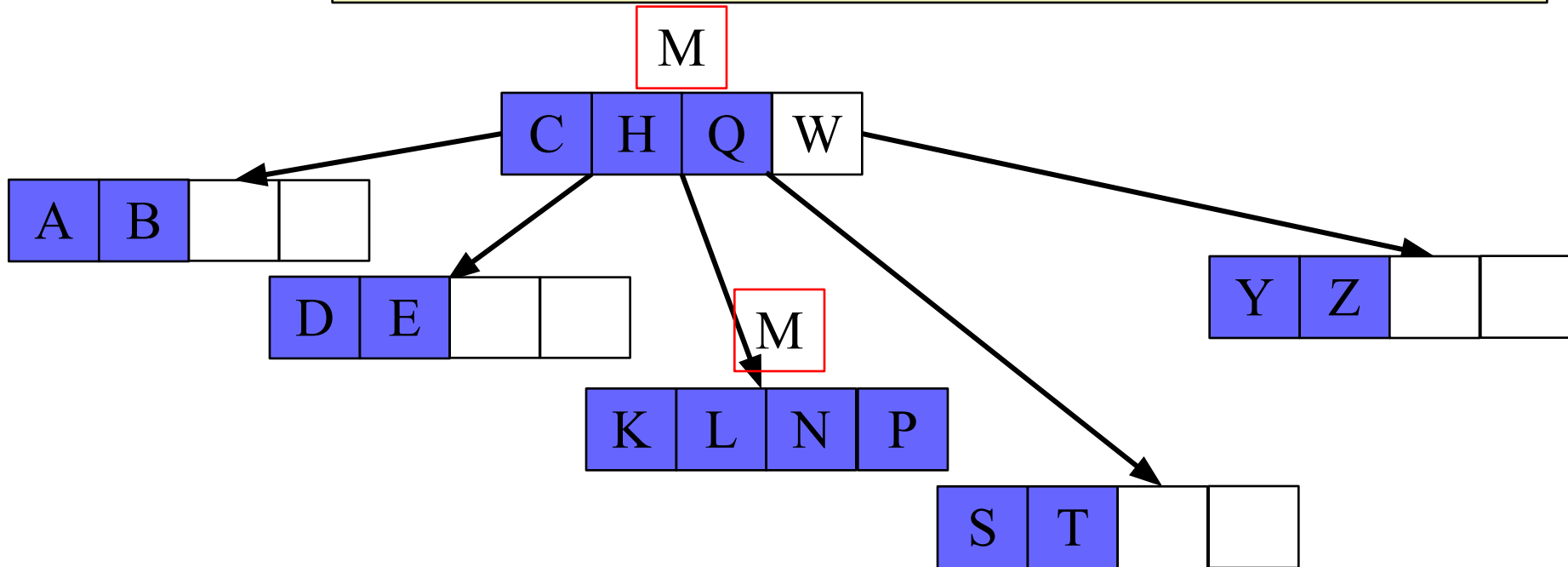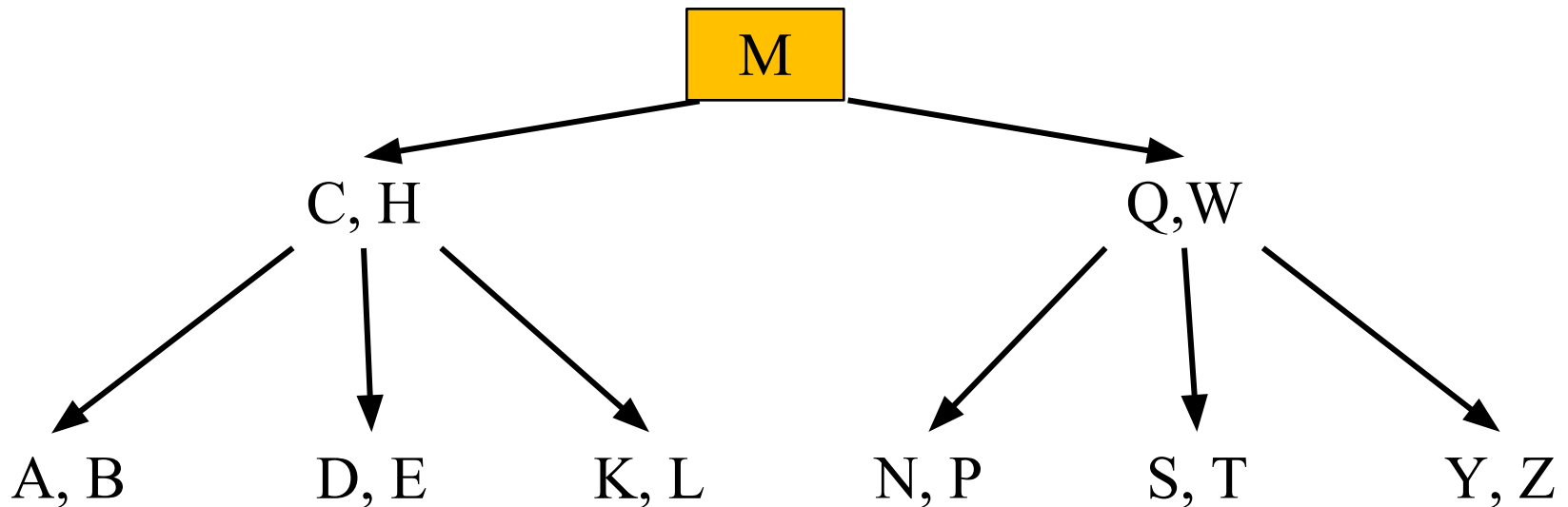| S | T | | |

| Y | Z | | |

Insertion in B-Tree

Create a B – tree of order 5 with the following set of data
D, H, Z, K, B, P, Q, E, A, S, W, T, C, L, N, Y, M

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

M

C, H

Q,W

A, B

D, E

K, L

N, P

S, T

Y, Z

Thank You