CSE-281: Data Structures and Algorithms

Trees
(Chapter-7)

Ref: Schaum's Outline Series, Theory and problems of Data Structures
By Seymour Lipschutz

And Online Resource

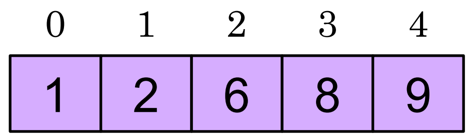
Eftekhar Hossain
Lecturer
Dept. of ETE, CUET

Topics to be Covered

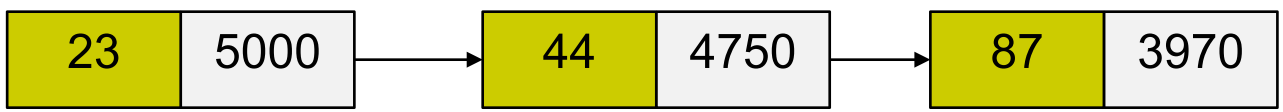
- Binary Tree
- Tree Traversal
- Binary Search Tree
- AVL Tree
- B-Tree

Introduction to Trees

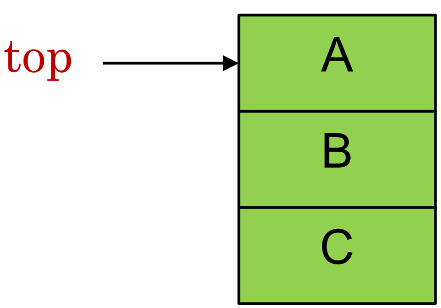
- Linear Data Structure



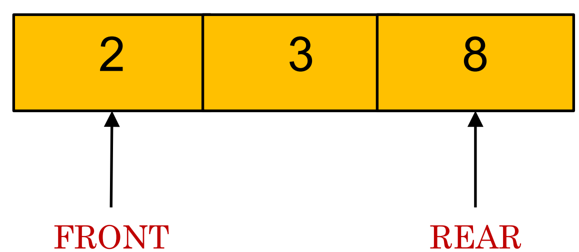
Array



Linked List



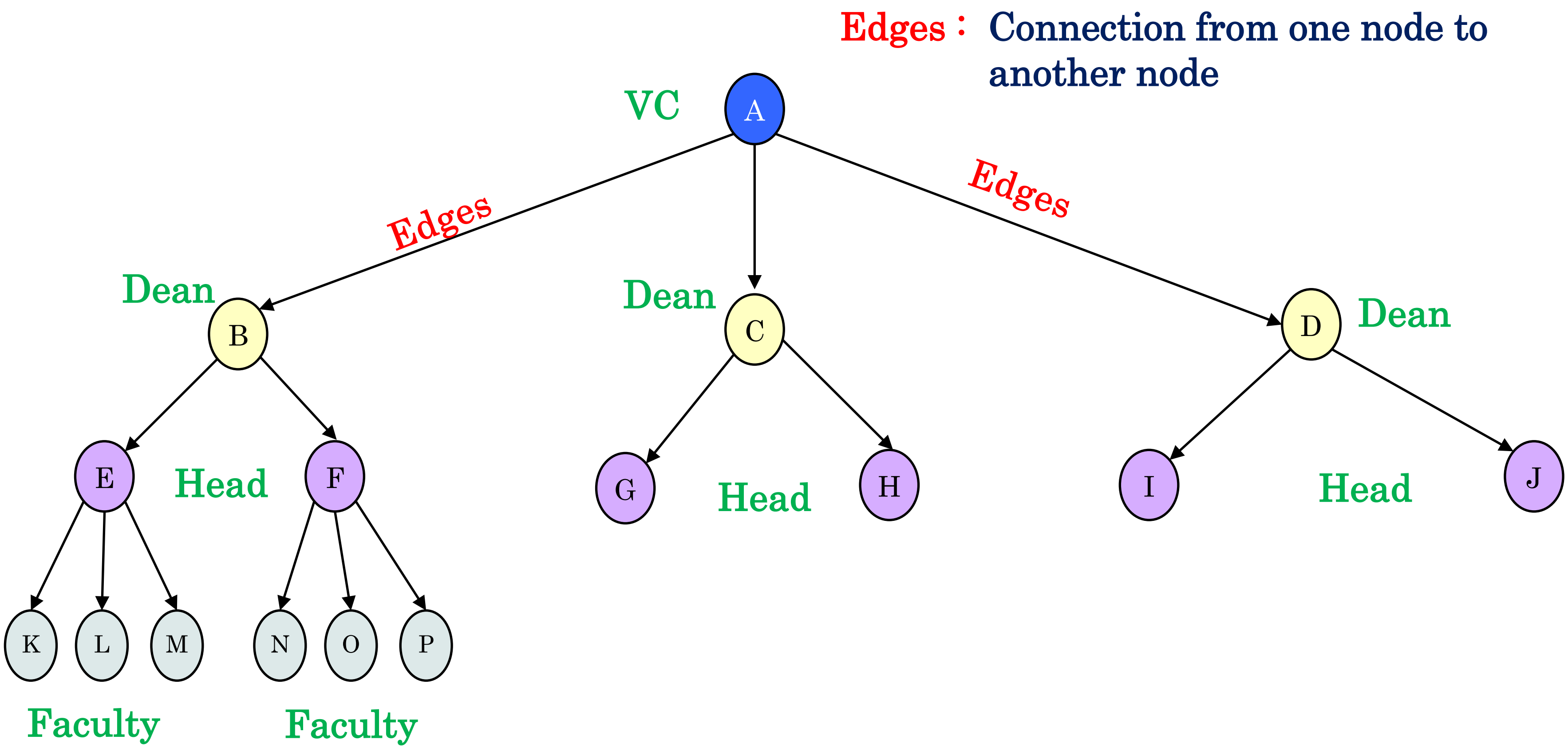
Stack



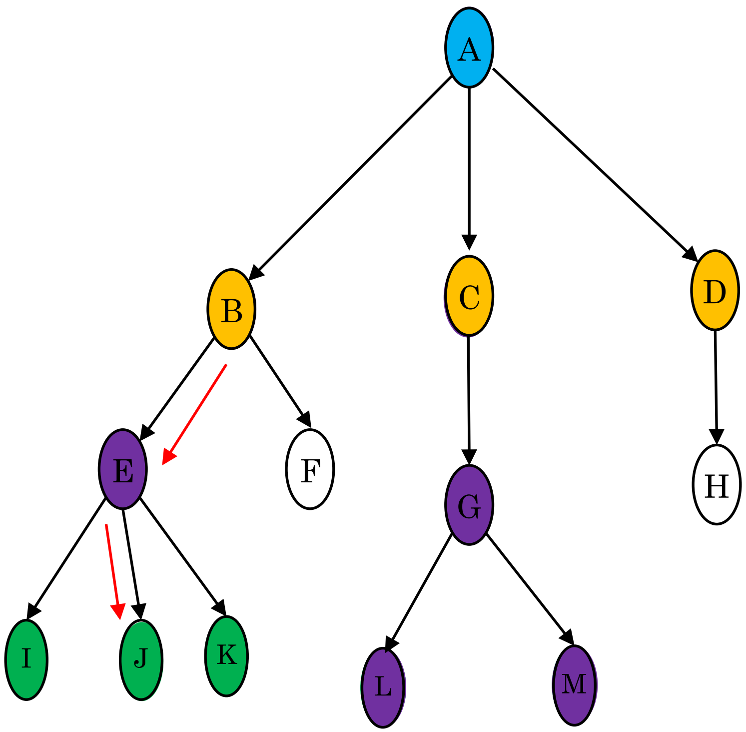
Queue

Introduction to Trees

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



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

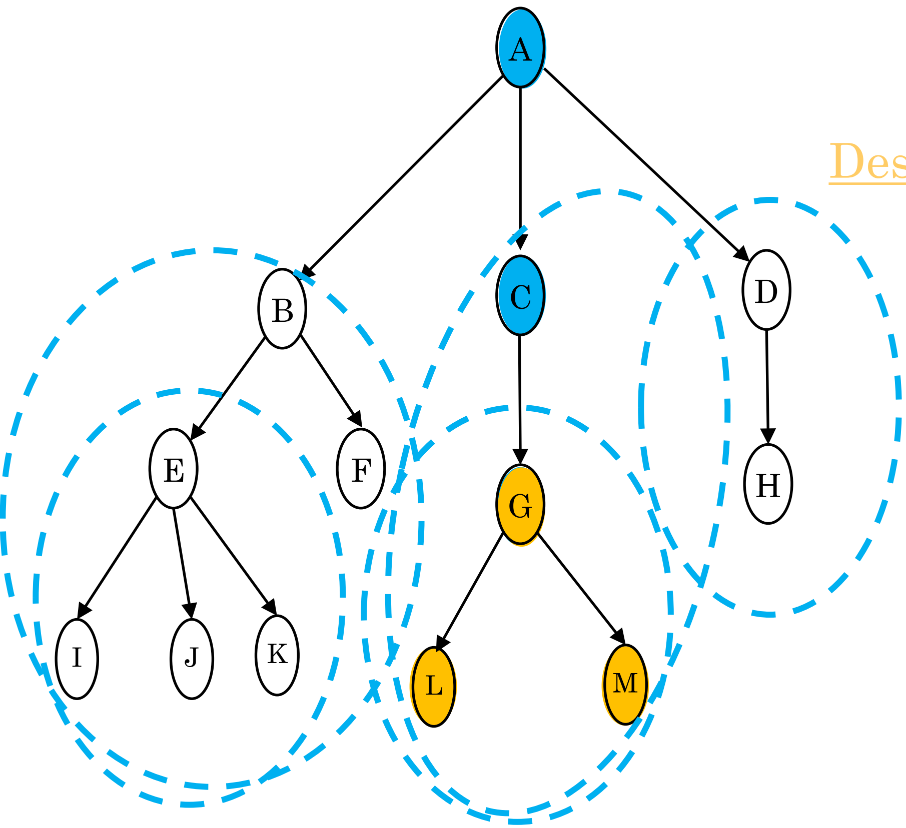
Non – Leaf nodes : have at least one children A , B, C, D, E, G

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 .

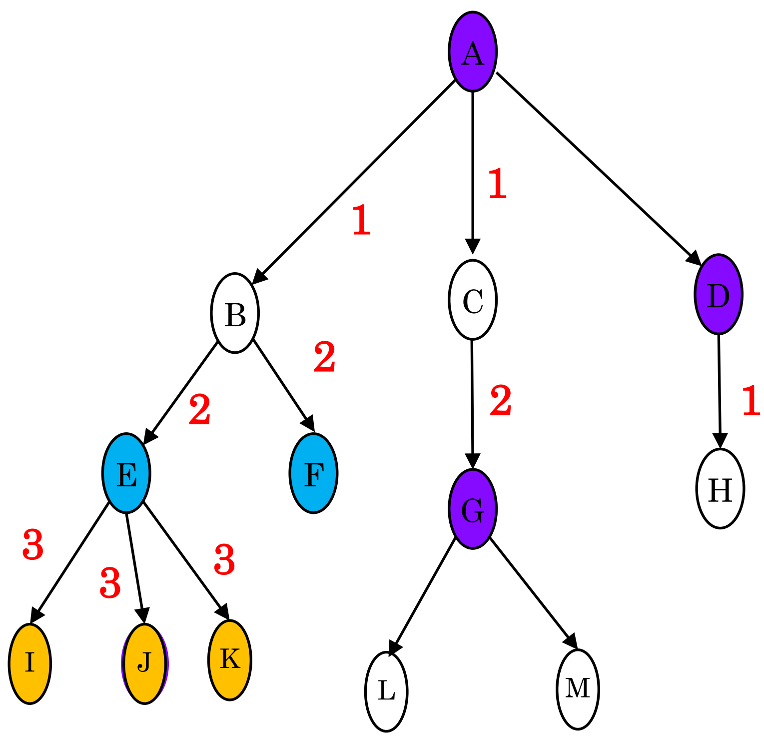
Ancestor of L → A, C, G

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



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

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

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

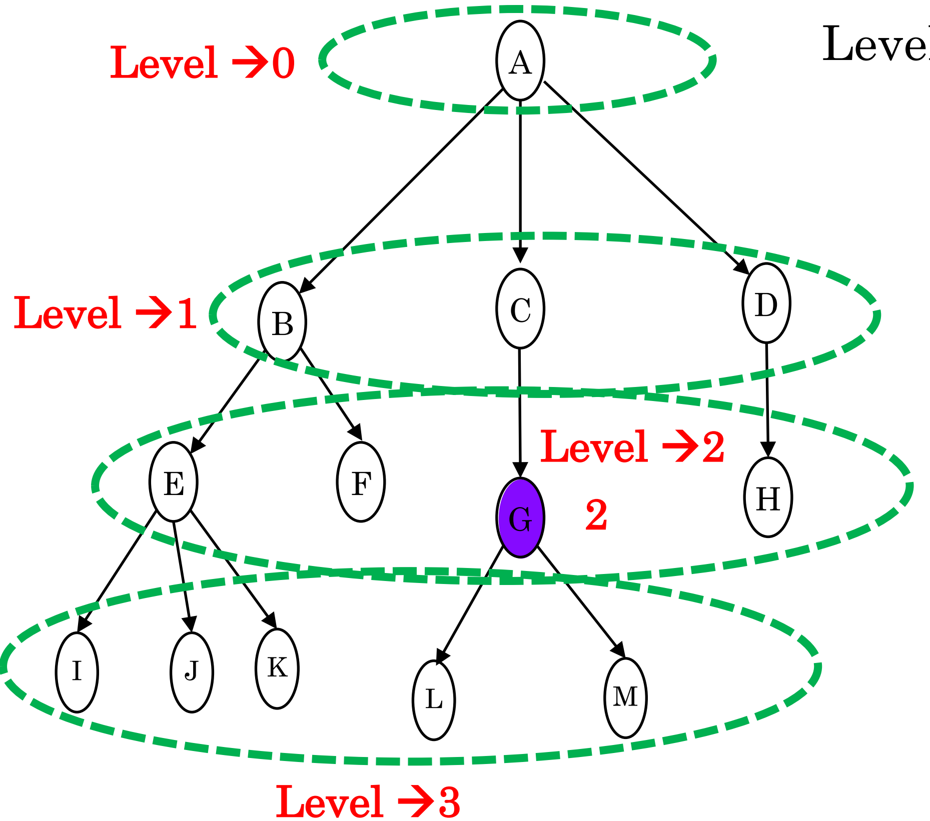
Depth : length of path from root to that node

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 → 2

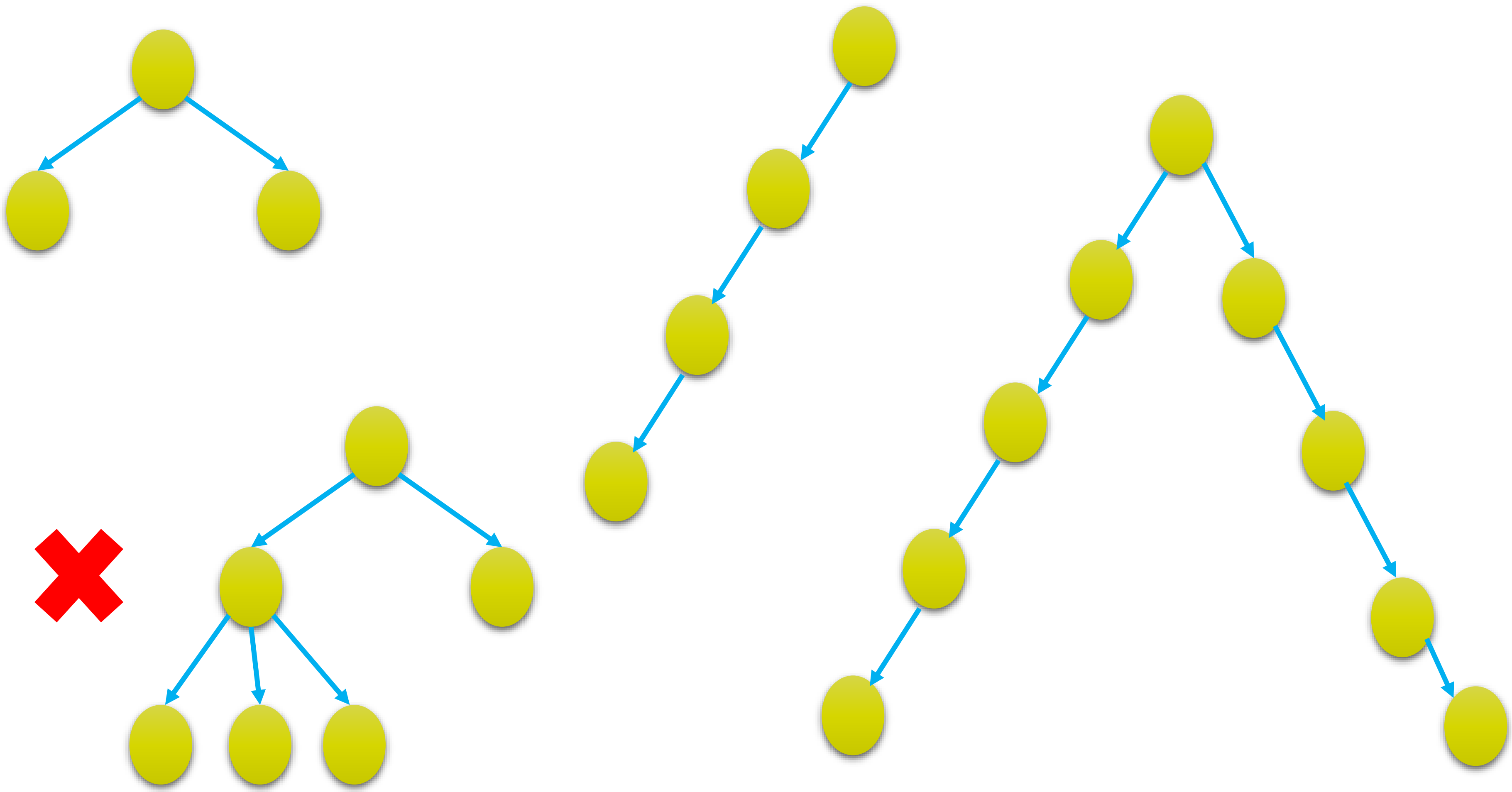
Level of a Tree is equal to the Height of the Tree , here it is 3

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

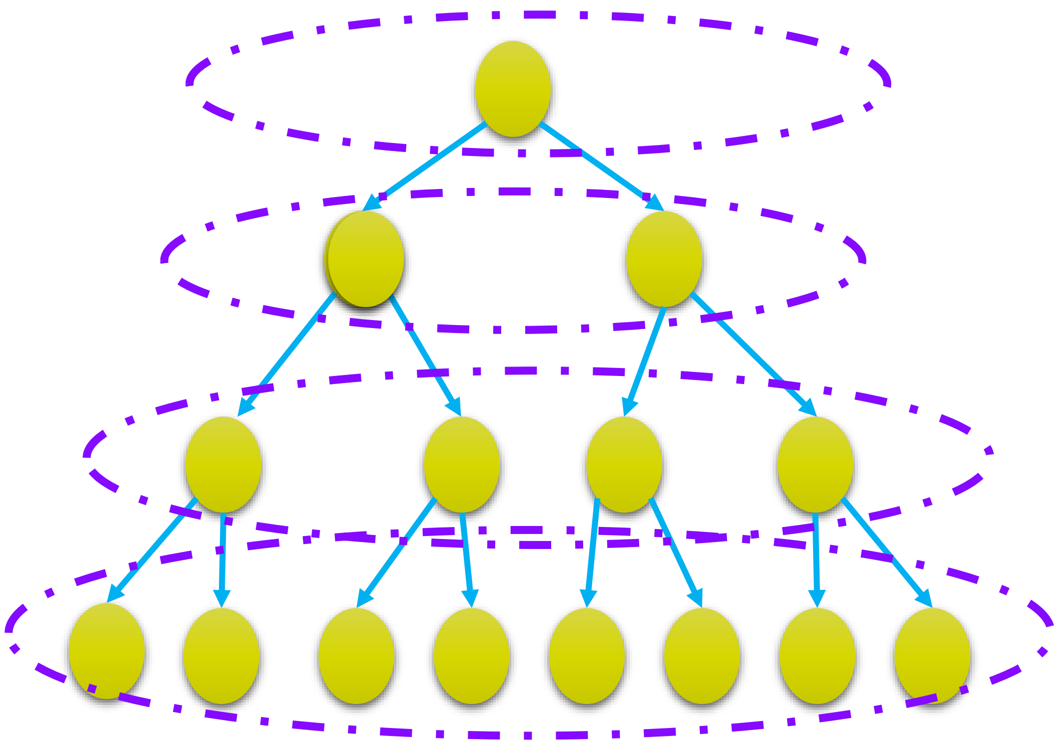
Binary Trees

- Each node have at most 2 children



Properties of Binary Trees

- At i^{th} level, a binary tree can have maximum 2^i nodes.



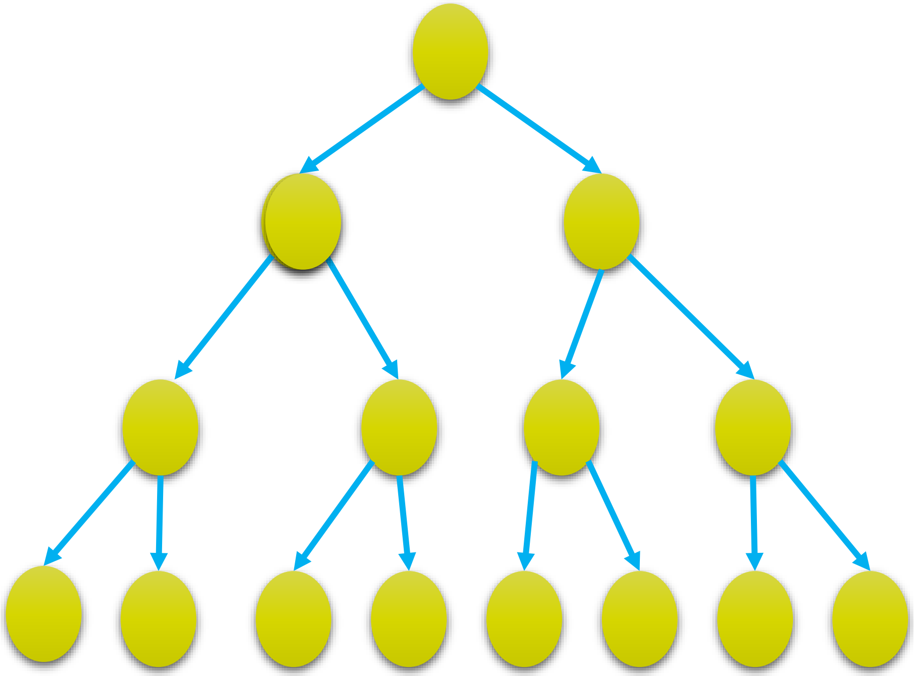
Level	Max Nodes	
0	1	2^0
1	2	2^1
2	4	2^2
3	8	2^3

Properties of Binary Trees

Max no. of nodes at height h

$= 2^0 + 2^1 + 2^2 + 2^3 + \ldots\ldots\ldots + 2^h$

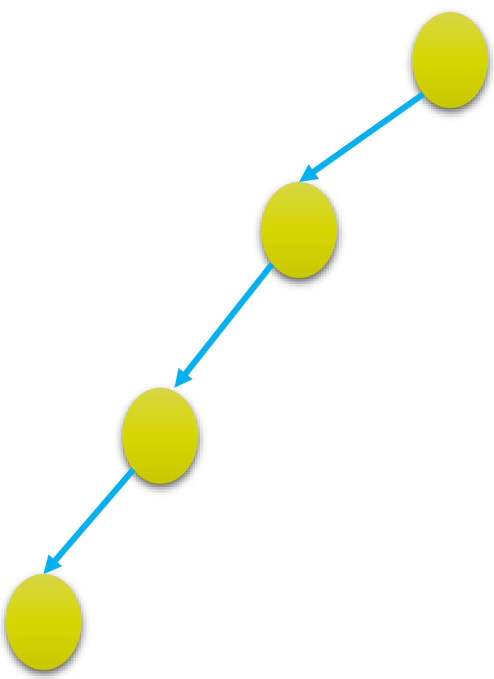
$= 2^{h+1} - 1$



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

Properties of Binary Trees

Min no. of nodes at height h	$= h + 1$



Level	Min Nodes
0	1
1	1
2	1
3	1

Properties of Binary Trees

Minimum height h $= \log_2(n+1) - 1$

Maximum height h $= n - 1$

Possible maximum height of the tree if n nodes are given

Possible minimum height of the tree if n nodes are given

$n = 2^{h+1} - 1$

$h = \log_2(n+1) - 1$

$n = h + 1$

$h = n - 1$

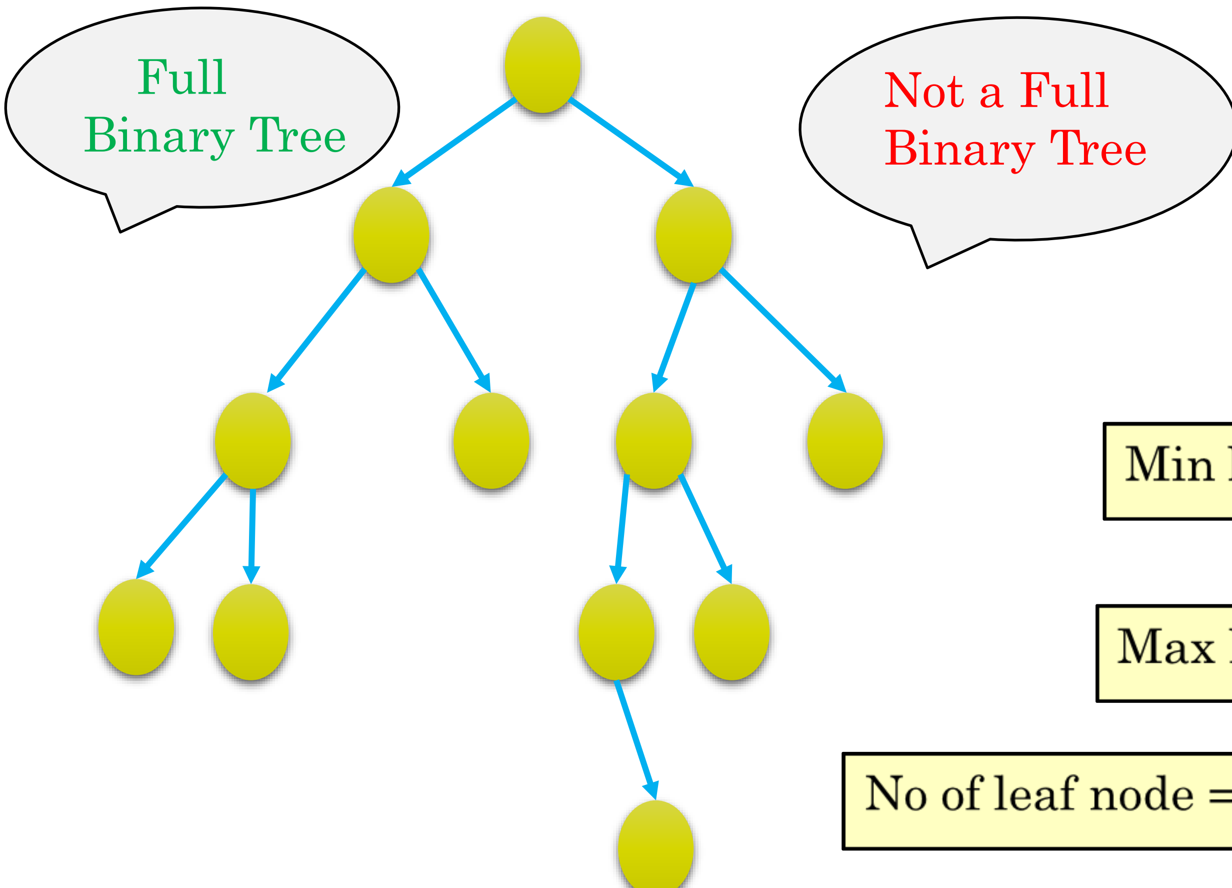
Types of Binary Trees

- Full / Proper / Strict
- Complete
- Perfect Binary Tree
- Degenerate Binary Tree

Full Binary Tree

Each node have either 0 or 2 children

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



Max nodes = $2^{h+1} - 1$

Min nodes = $2h + 1$

Min height h = $\log_2(n+1) - 1$

Max height h = $n - 1/2$

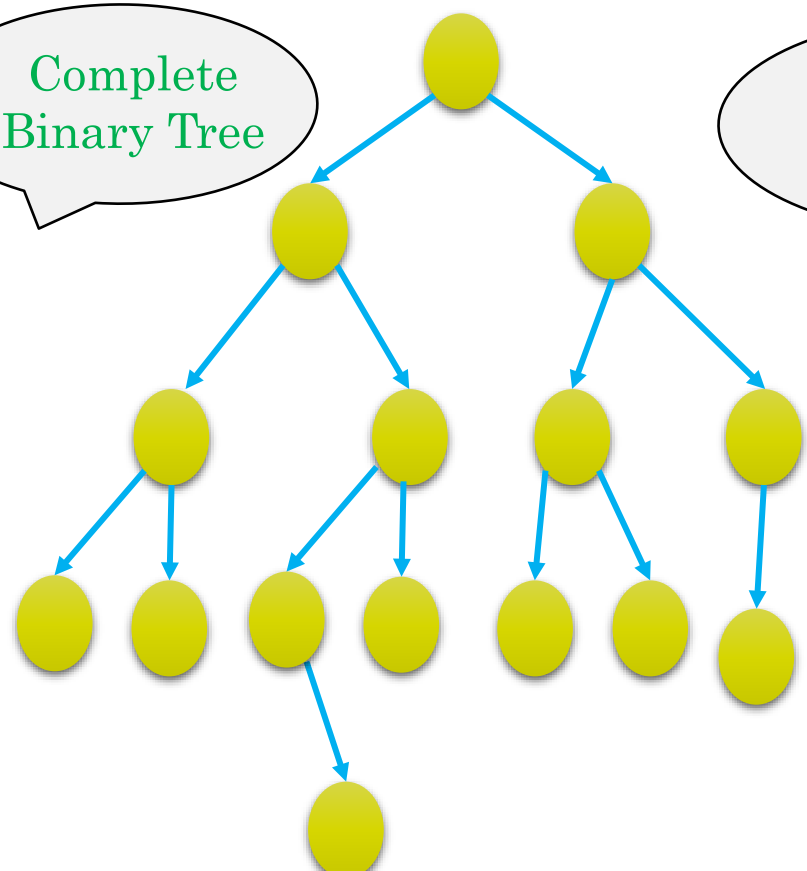
No of leaf node = $no\ of\ internal\ nodes + 1$

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

$\text{Max nodes} = 2^{h+1} - 1$

$\text{Min nodes} = 2^{h}$

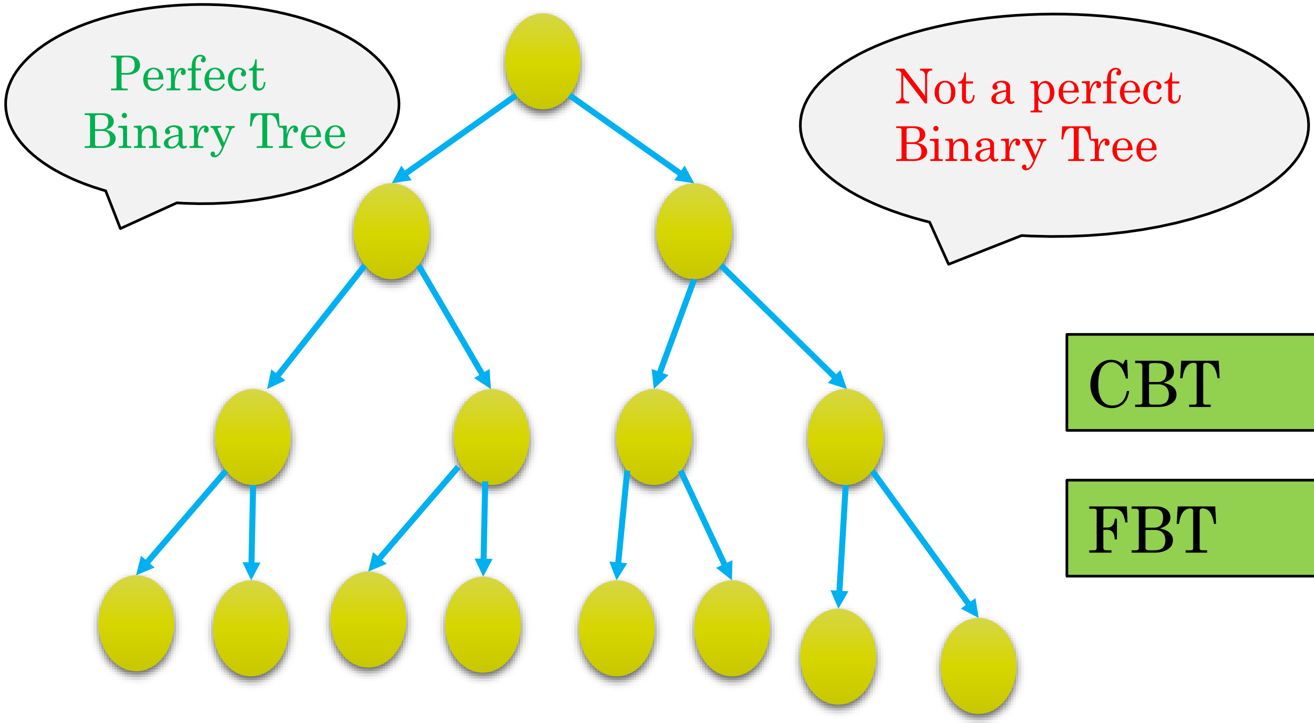
$\text{Min height } h = \log_2(n+1) - 1$

$\text{Max height h} = \log_2 n$

Perfect Binary Trees

All internal nodes have 2 children

And all leaf nodes are at same level



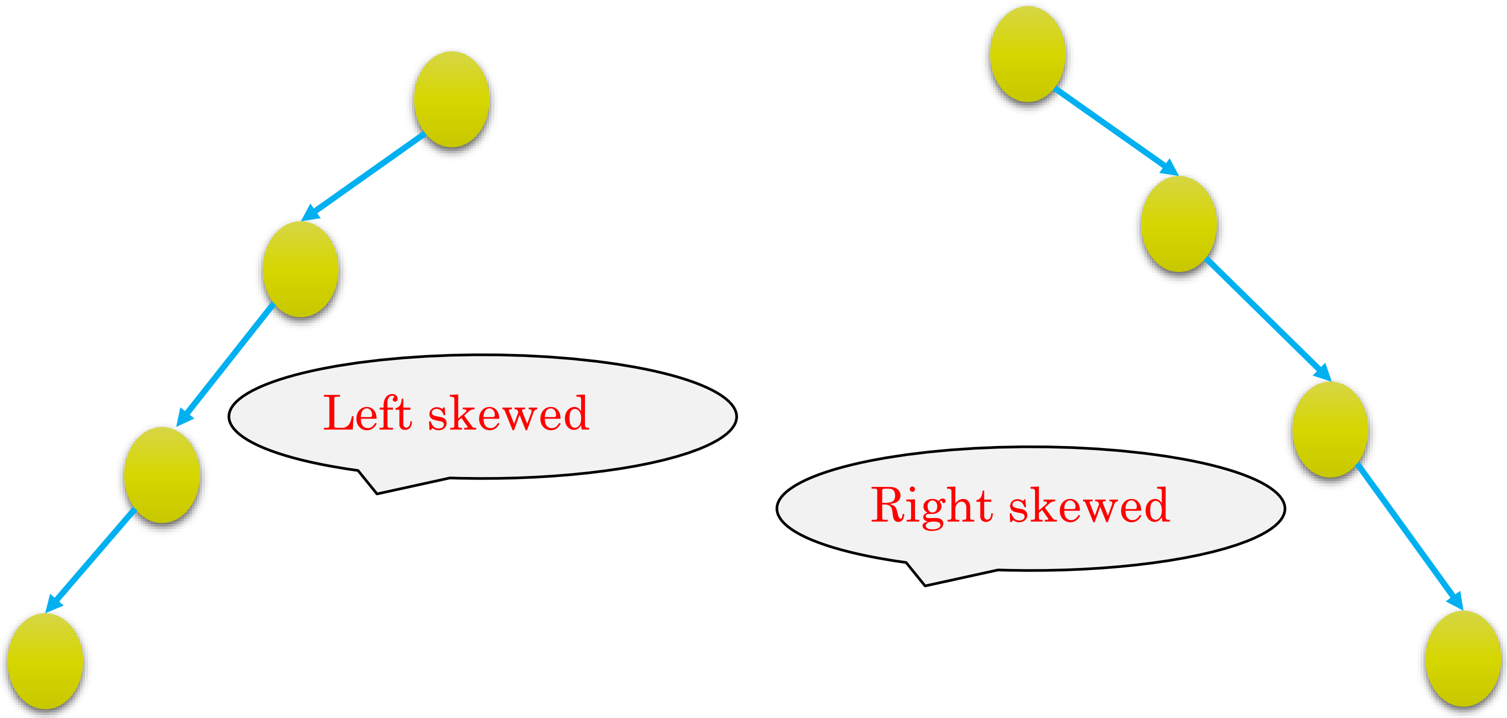
CBT

FBT

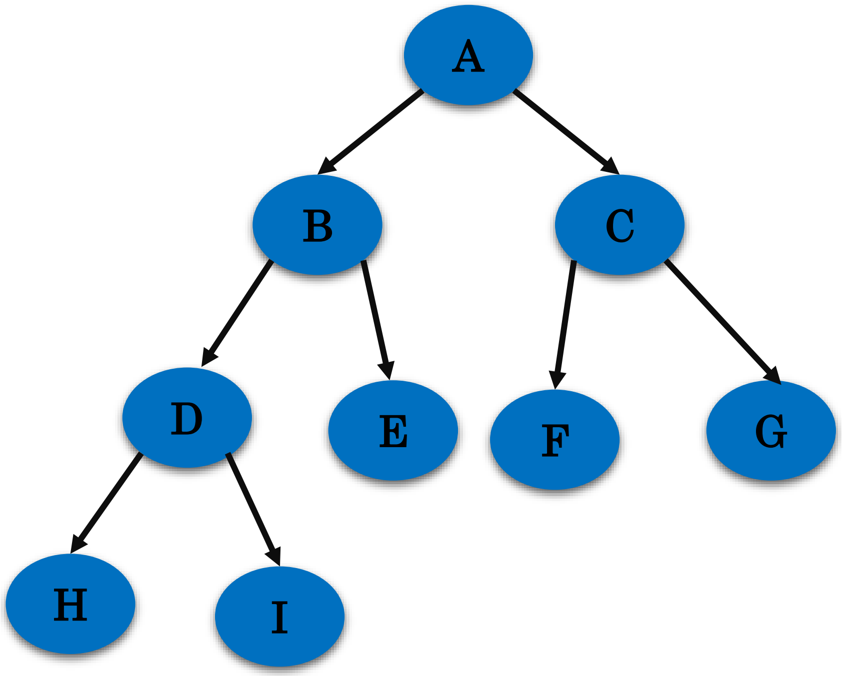
A PBT can be a CBT or FBT or both

Degenerate Binary Trees

All internal nodes have only one children



Array Representation of BT

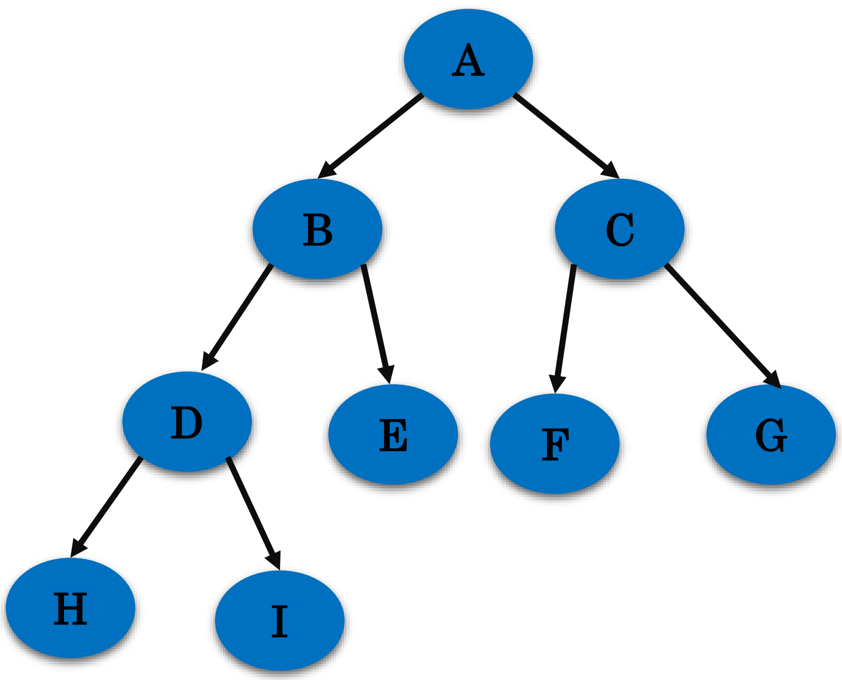


Fill up the array using the level
Of the tree from left to right

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 node is at i^{th} index :

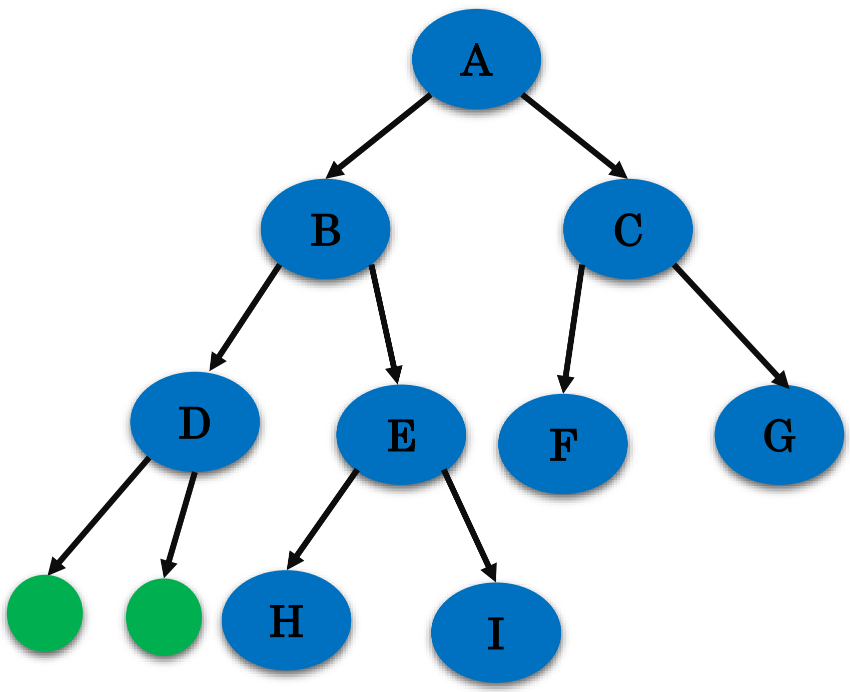
→ Left child would be at : $(2 * i) + 1$

→ Right child would be at : $(2 * i) + 2$

→ Parent would be at : $\lfloor \frac{(i-1)}{2} \rfloor$

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

Array Representation of BT



If a node is at i^{th} index :

→ Left child would be at : $(2 * i) + 1$

→ Right child would be at : $(2 * i) + 2$

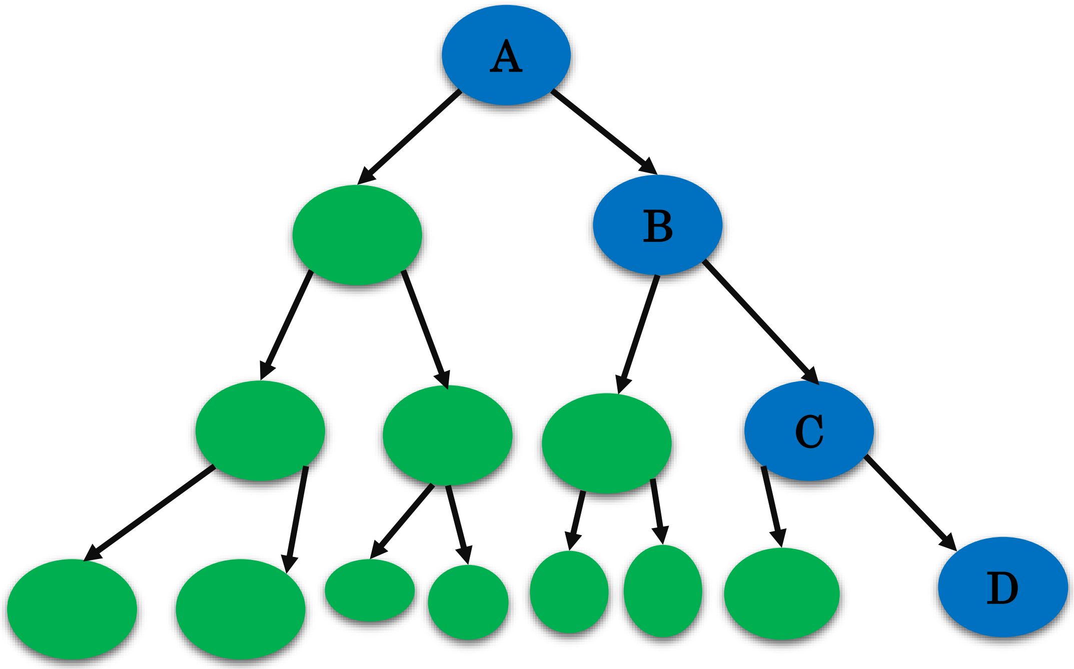
→ Parent would be at : $\frac{(i-1)}{2} \rfloor$

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.

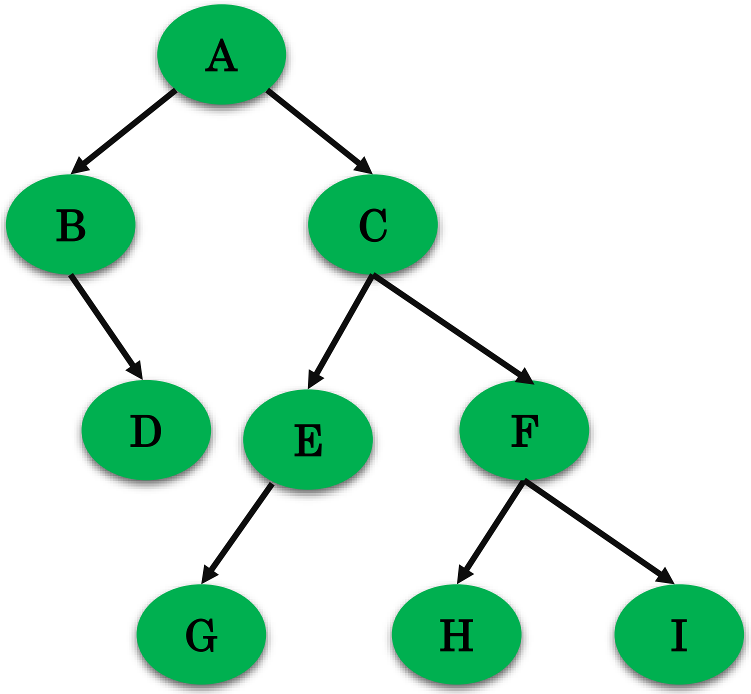
Inorder: Left Root Right

Preorder: Root Left Right

Postorder: Left Right Root

Inorder Traversal

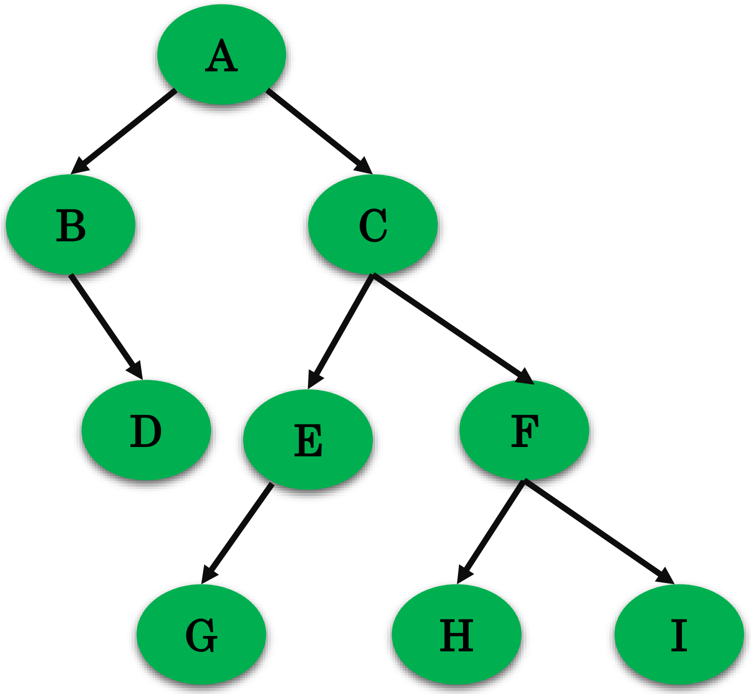
Inorder: Left Root Right



B	D	A	G	E	C	H	F	I

Preorder Traversal

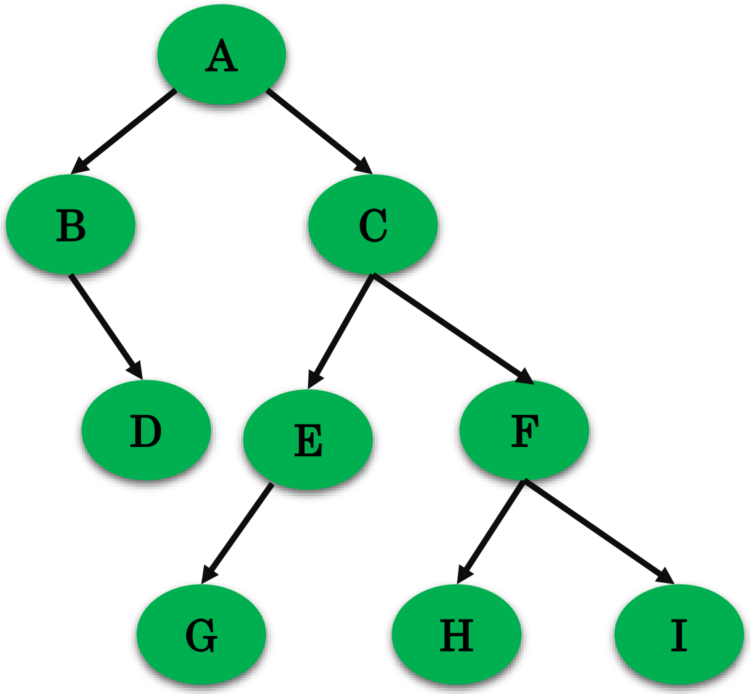
Preorder: Root Left Right



A	B	D	C	E	G	F	H	I

Postorder Traversal

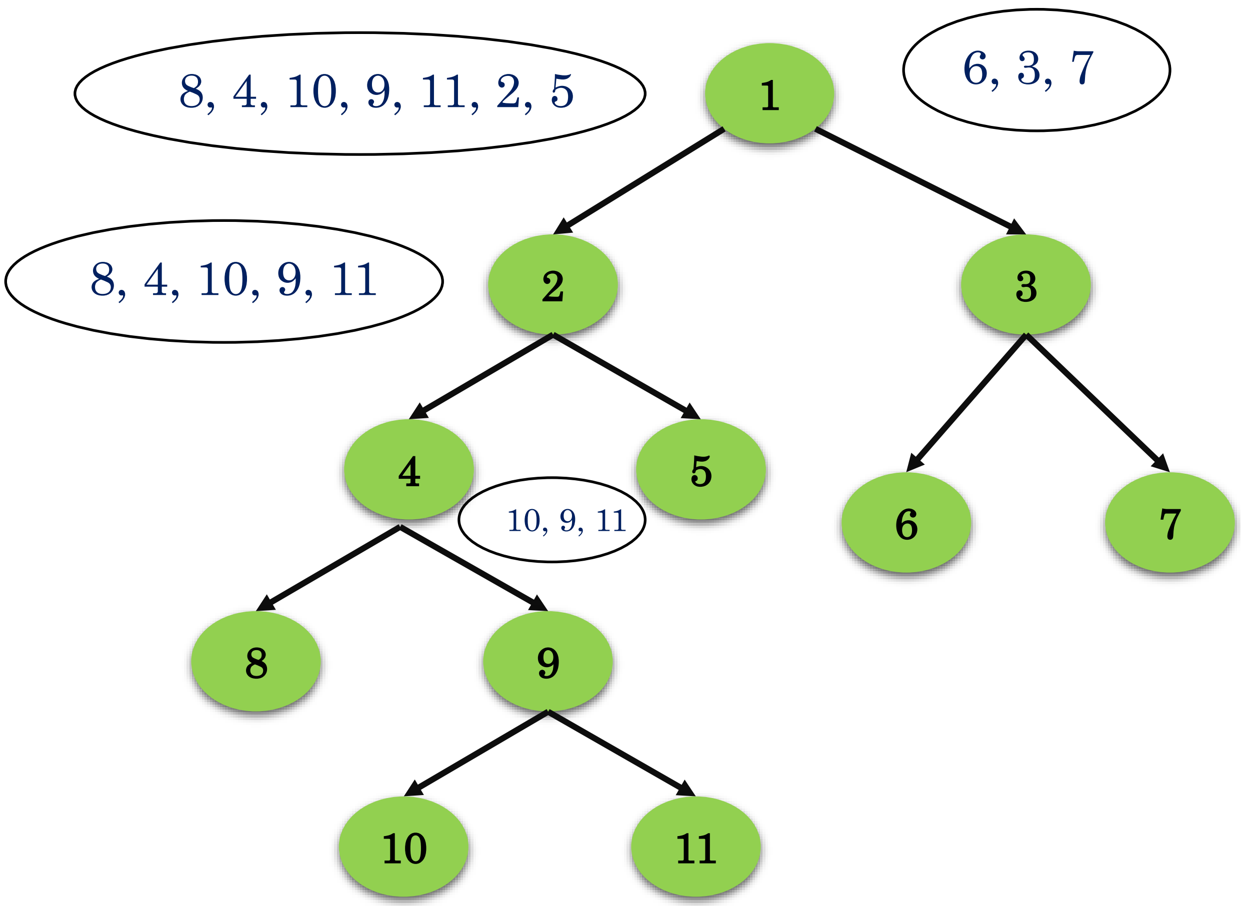
Postorder: Left Right Root



D	B	G	E	H	I	F	C	A

Construct Binary Tree

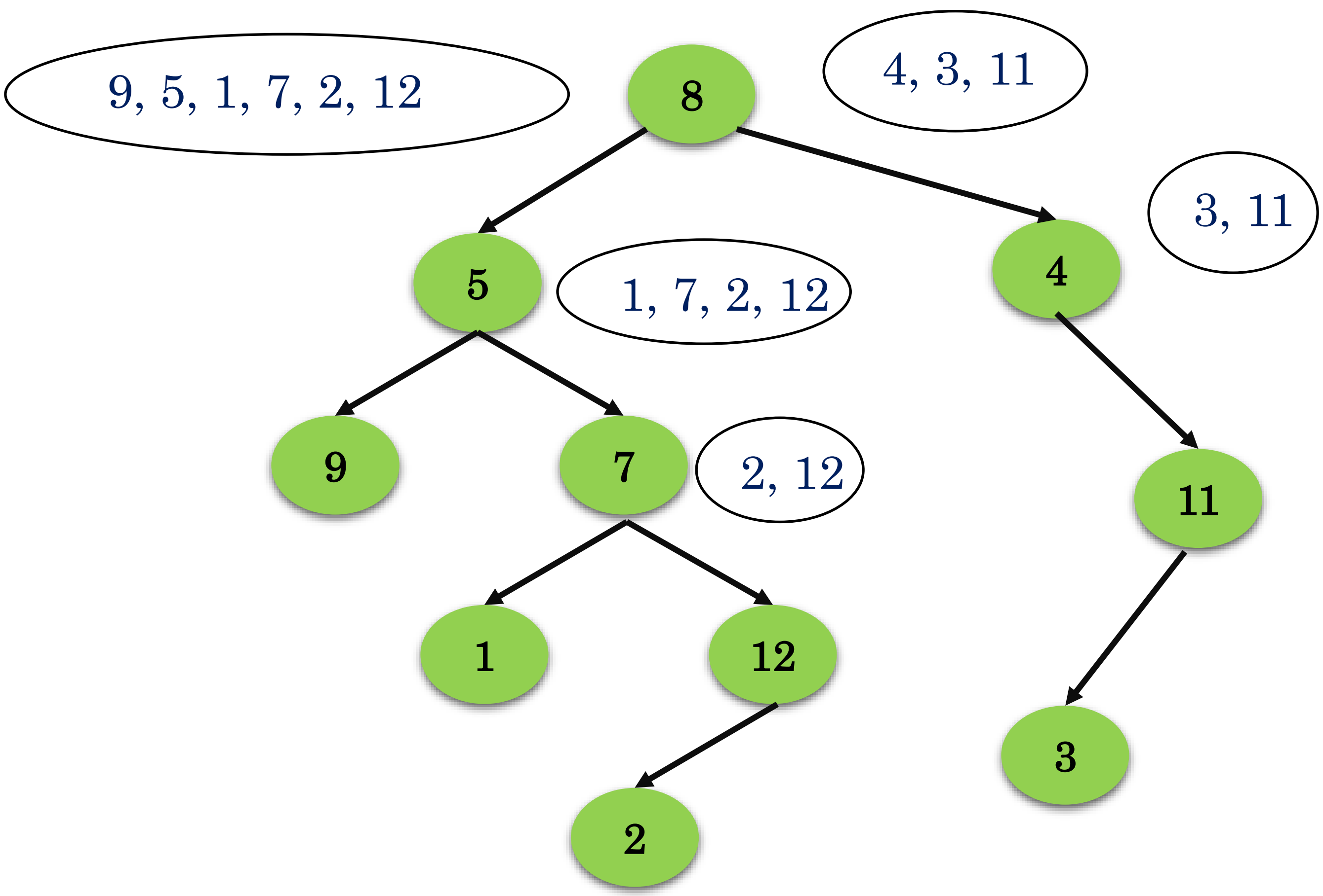
Preorder:	1	2	4	8	9	10	11	5	3	6	7	(Ro L R)
Inorder:	8	4	10	9	11	2	5	1	6	3	7	(L Ro R)



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)

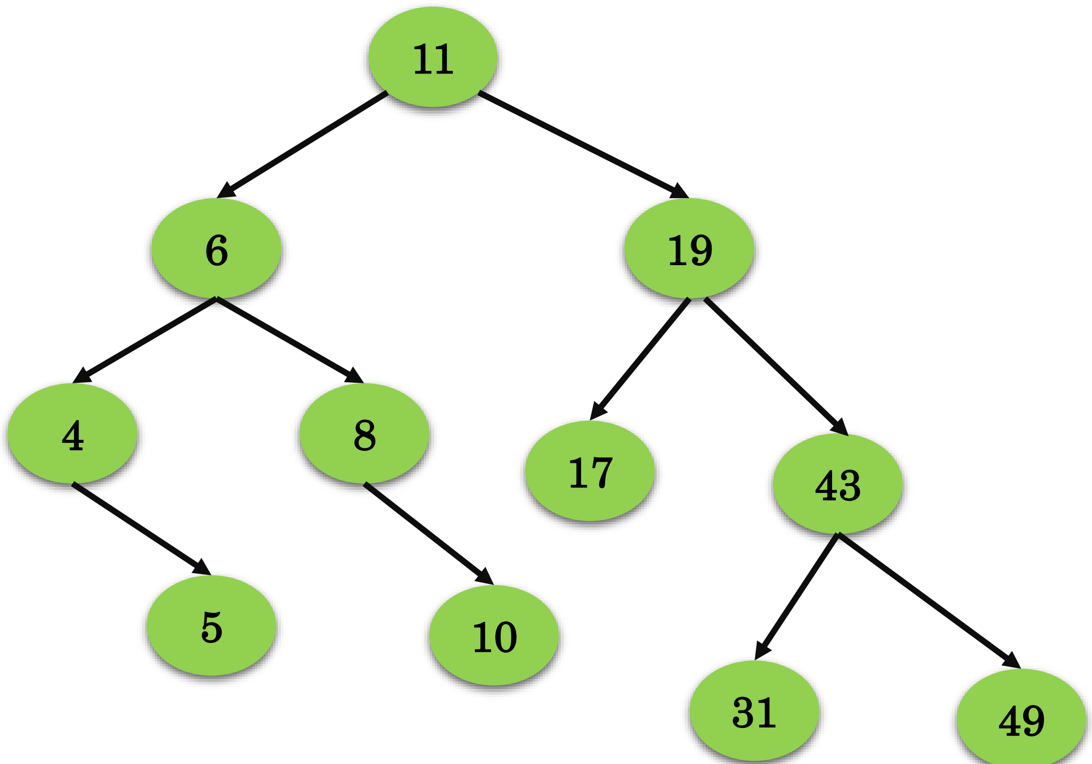


Binary Search Tree (BST)

Left subtree of a node contain values less than that node

Right subtree of a node contain values greater than that node

11 6 8 19 4 10 5 17 43 49 31



BST Complexity

Insertion
Deletion
Searching

Best Case → $O(1)$
Avg. Case → $O\ (\log n)$
Worst Case → $O\ (n)$

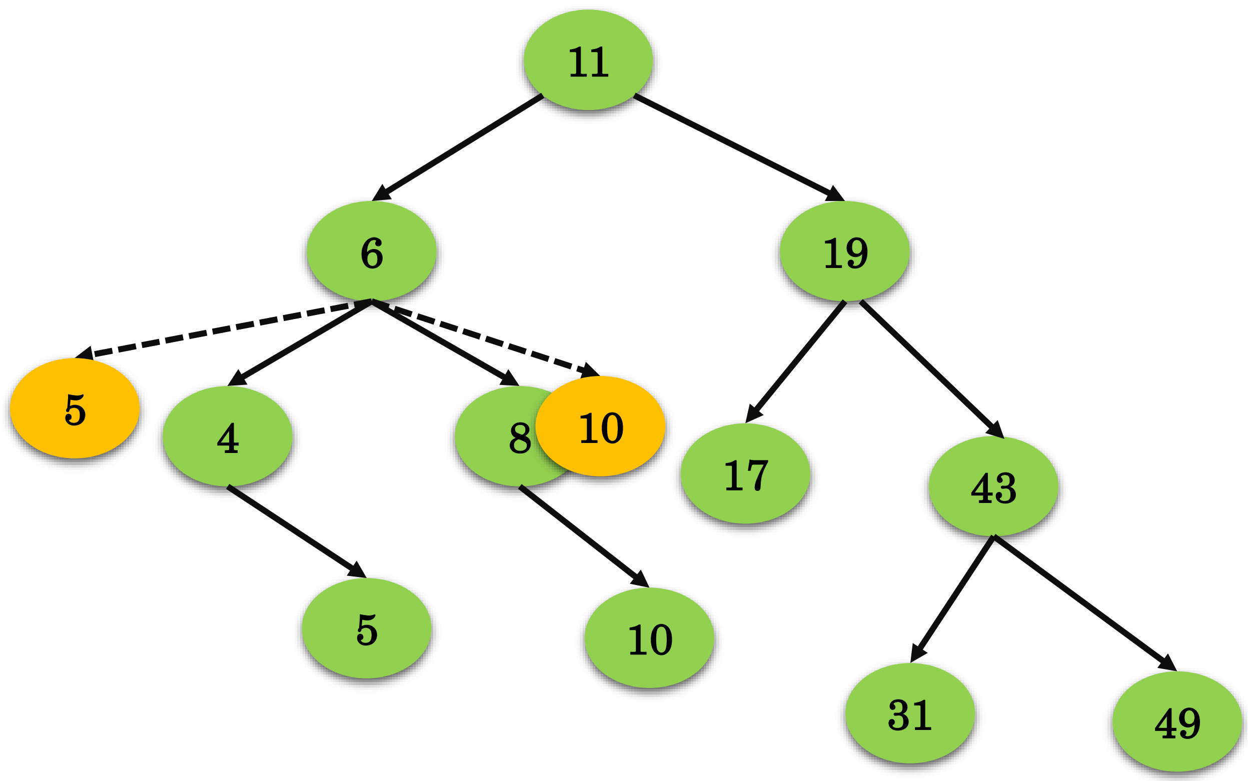
Binary Search Tree (BST)

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

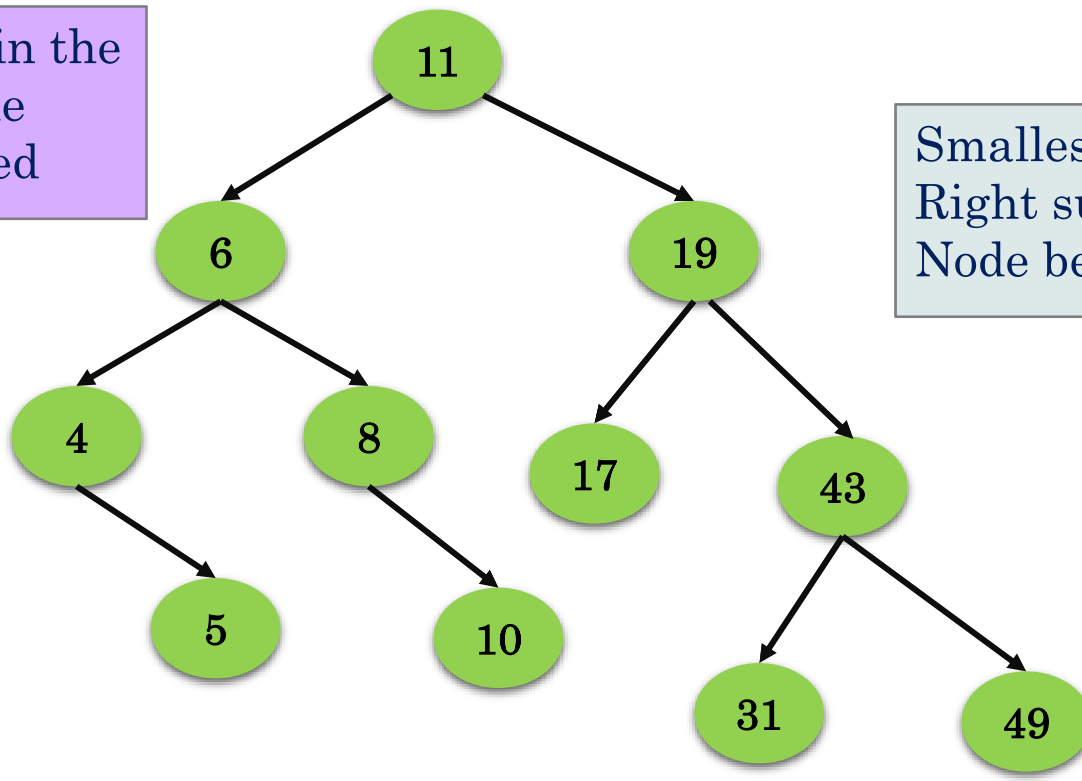


Binary Search Tree (BST)

Deletion: The node you want to delete may have

2 Children → Inorder predecessor → Inorder successor

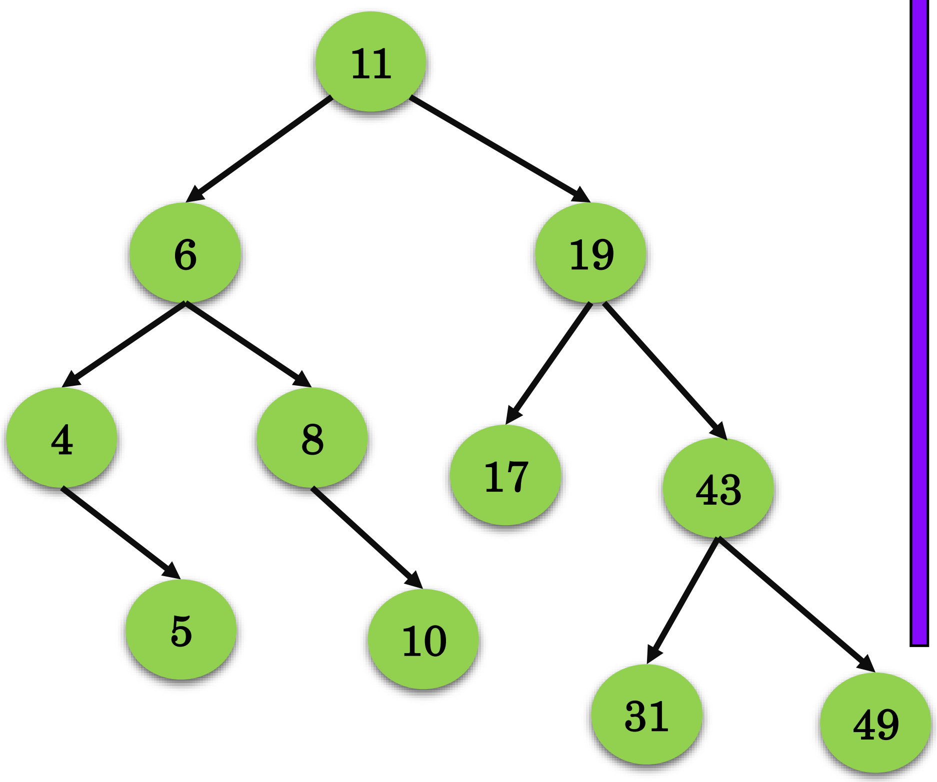
Largest element in the Left subtree of the Node being deleted



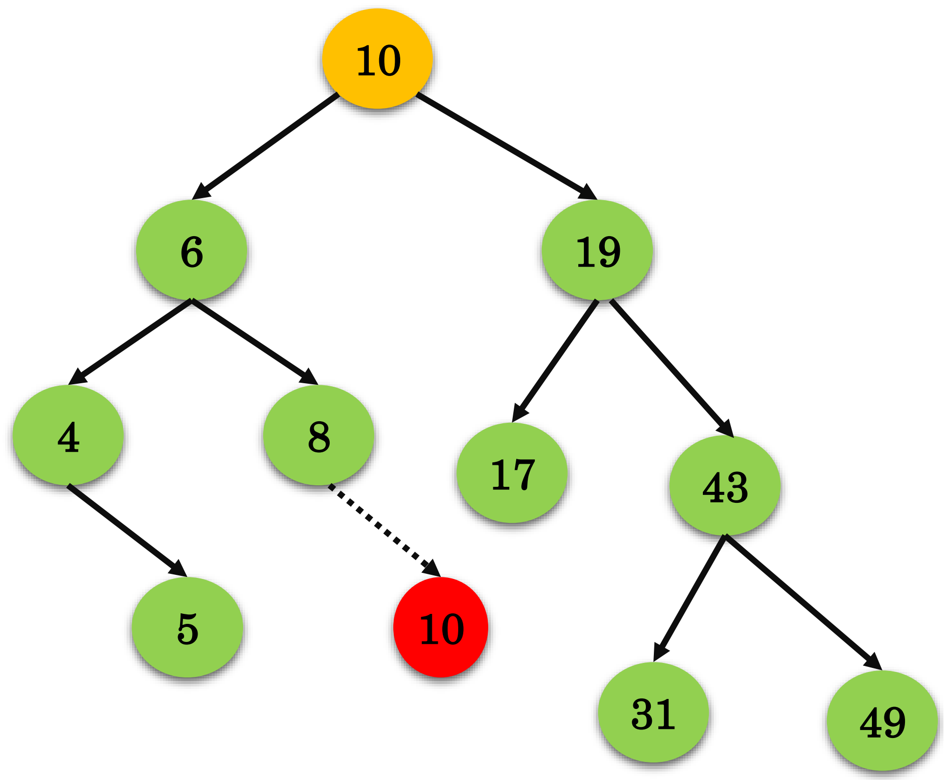
Smallest element in the Right subtree of the Node being deleted

Binary Search Tree (BST)

Suppose, we want to delete node 11



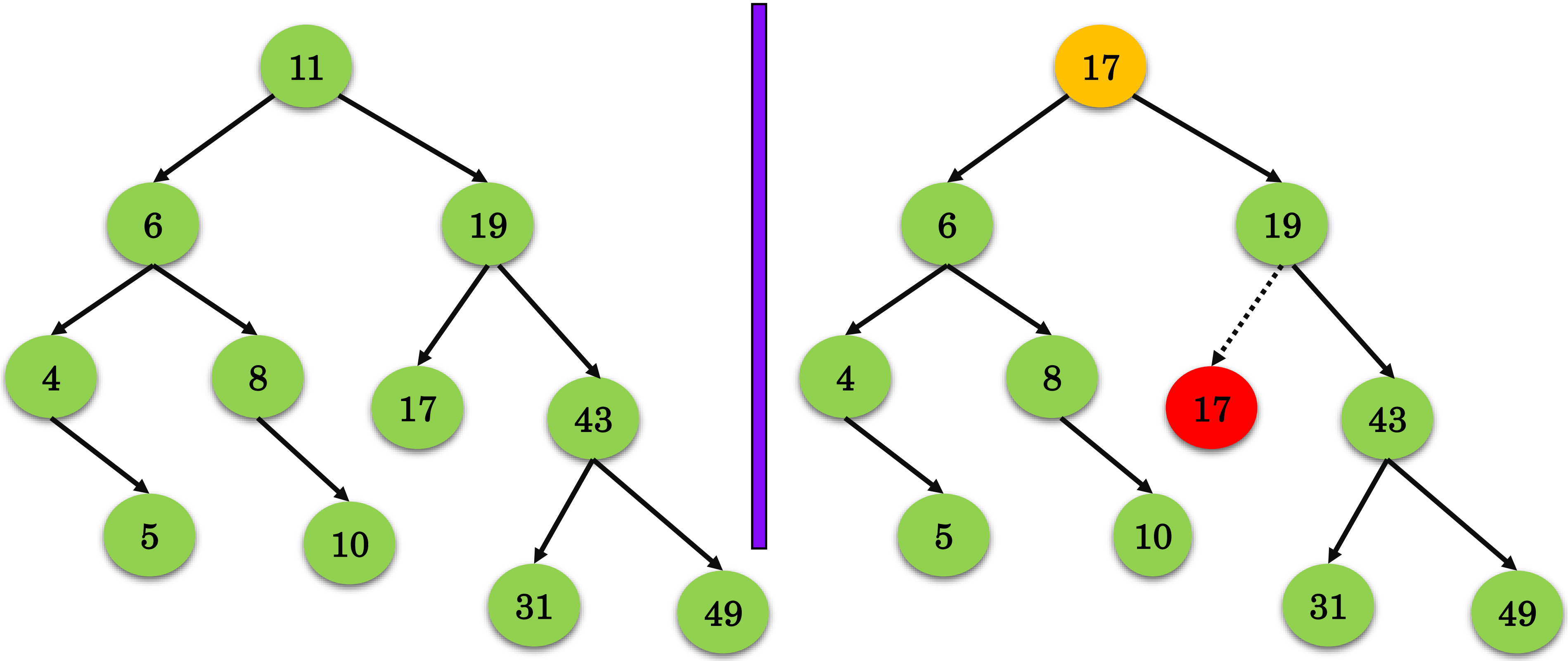
→ Inorder predecessor



Binary Search Tree (BST)

Suppose, we want to delete node 11

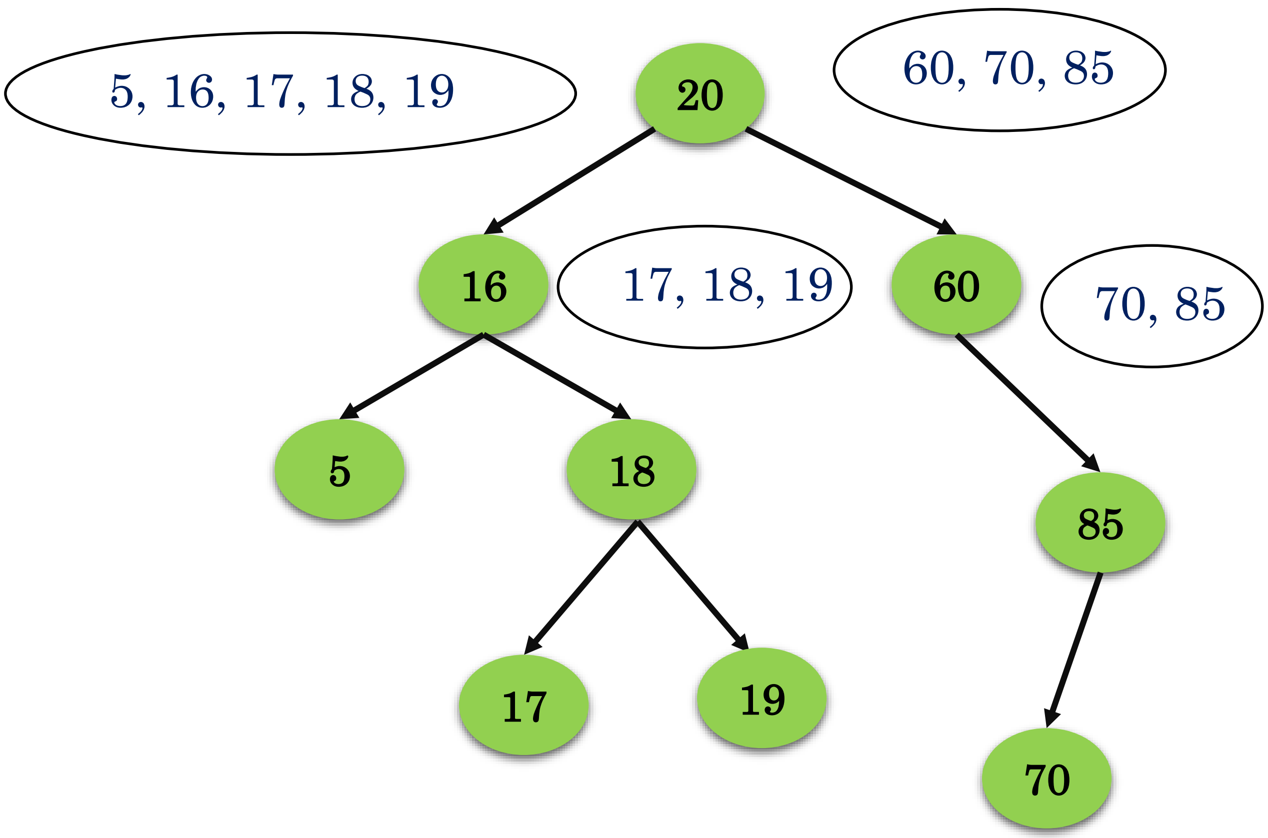
→ Inorder successor



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

i. It is a BST

ii. height of left subtree – height of right subtree = $\{1, 0, -1\}$

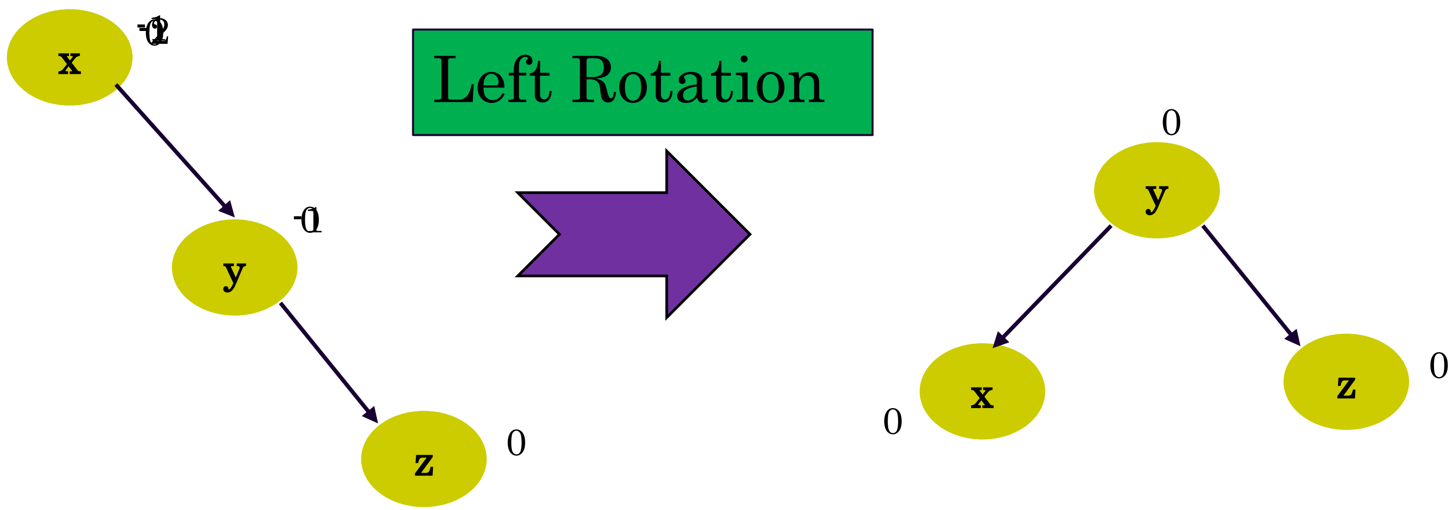
Balance Factor

AVL tree is a self balancing binary search tree

To balance a binary search tree four situations would arise

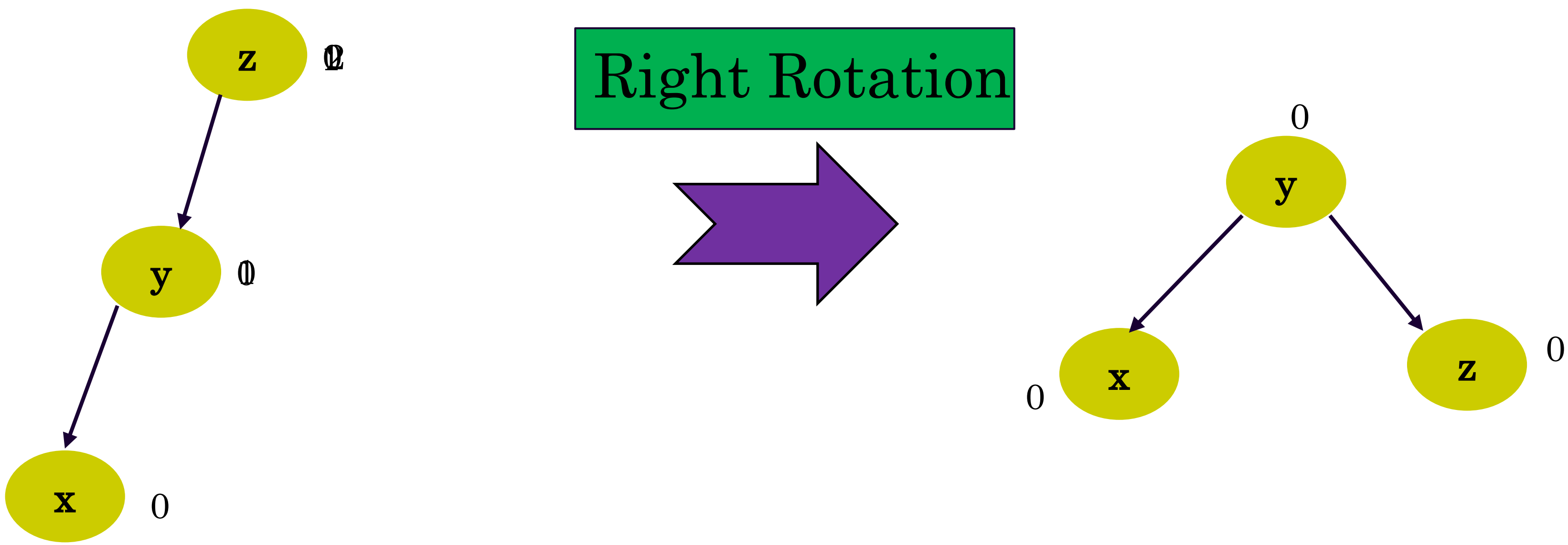
AVL Tree

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



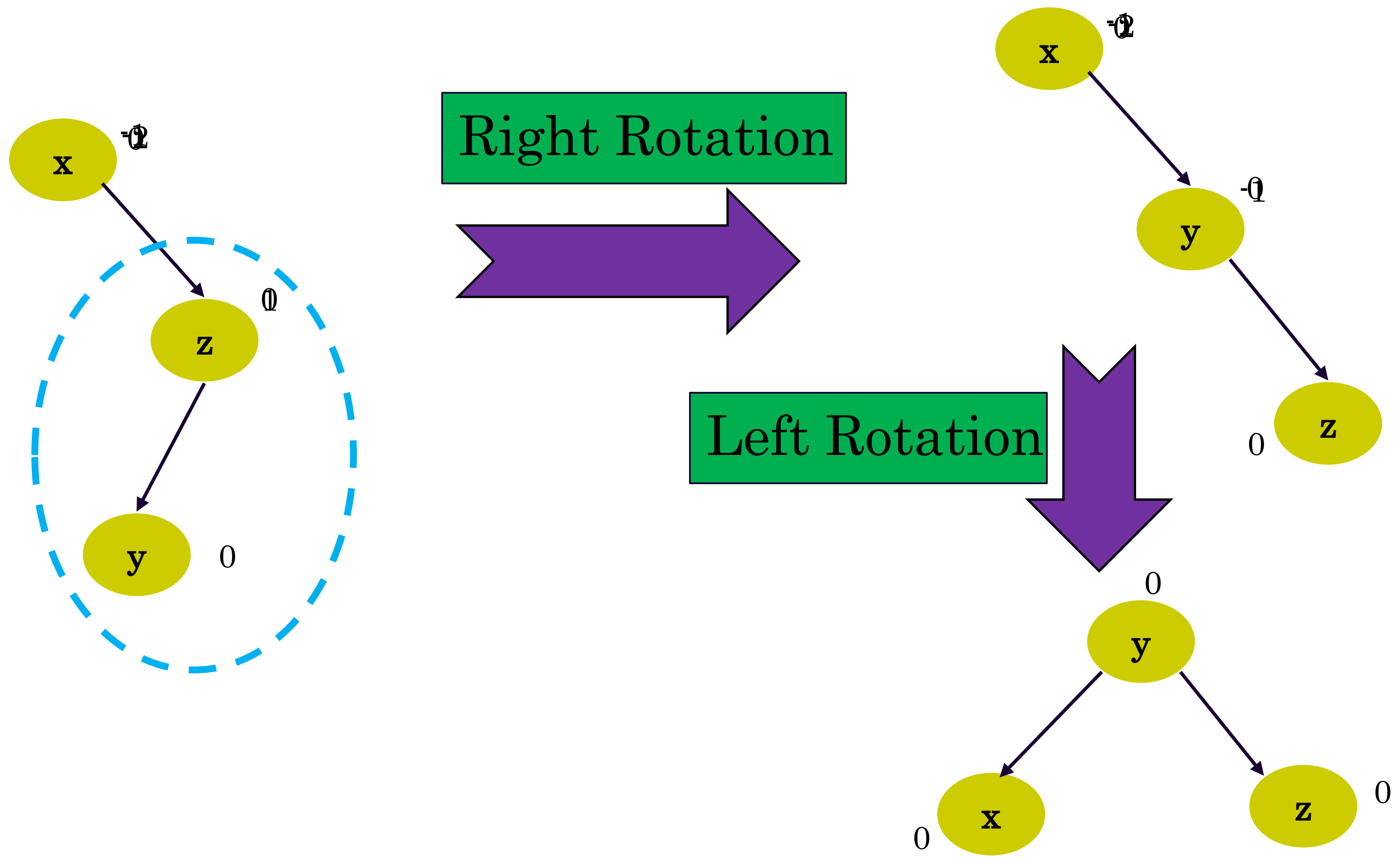
AVL Tree

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



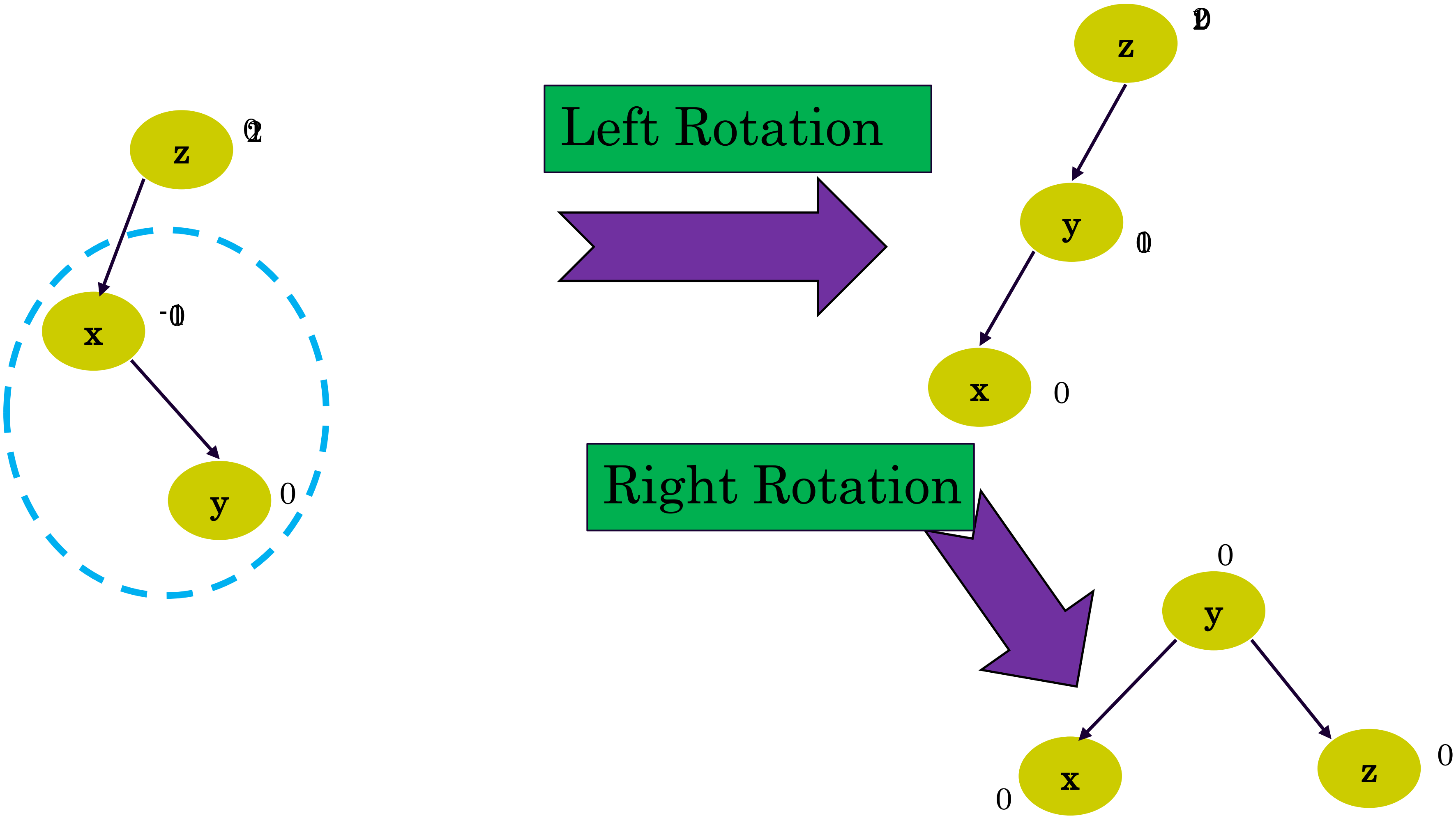
AVL Tree

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



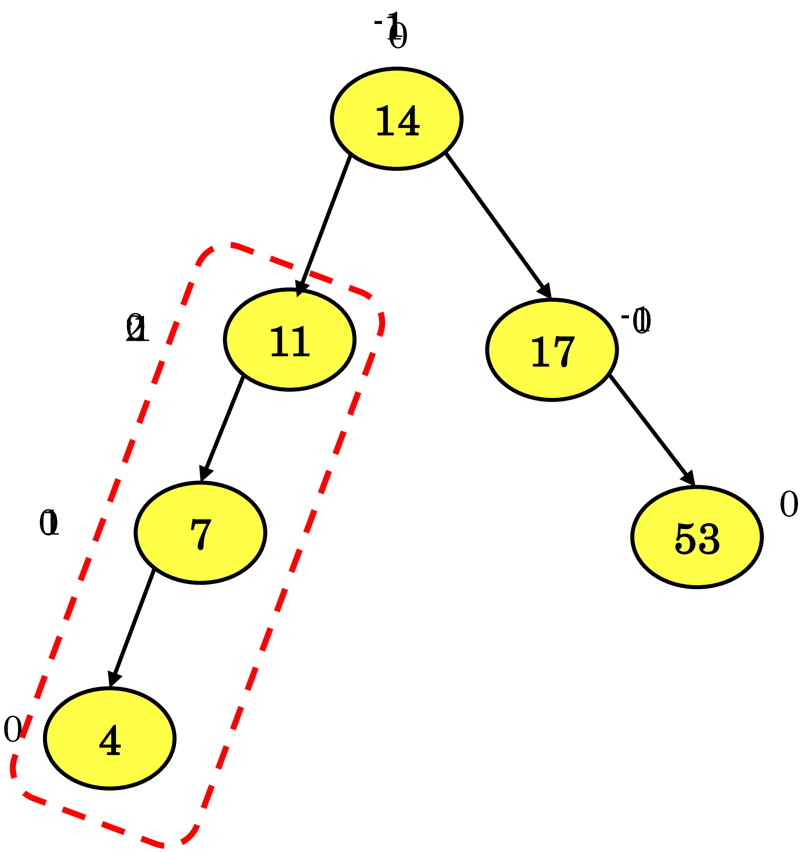
AVL Tree

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



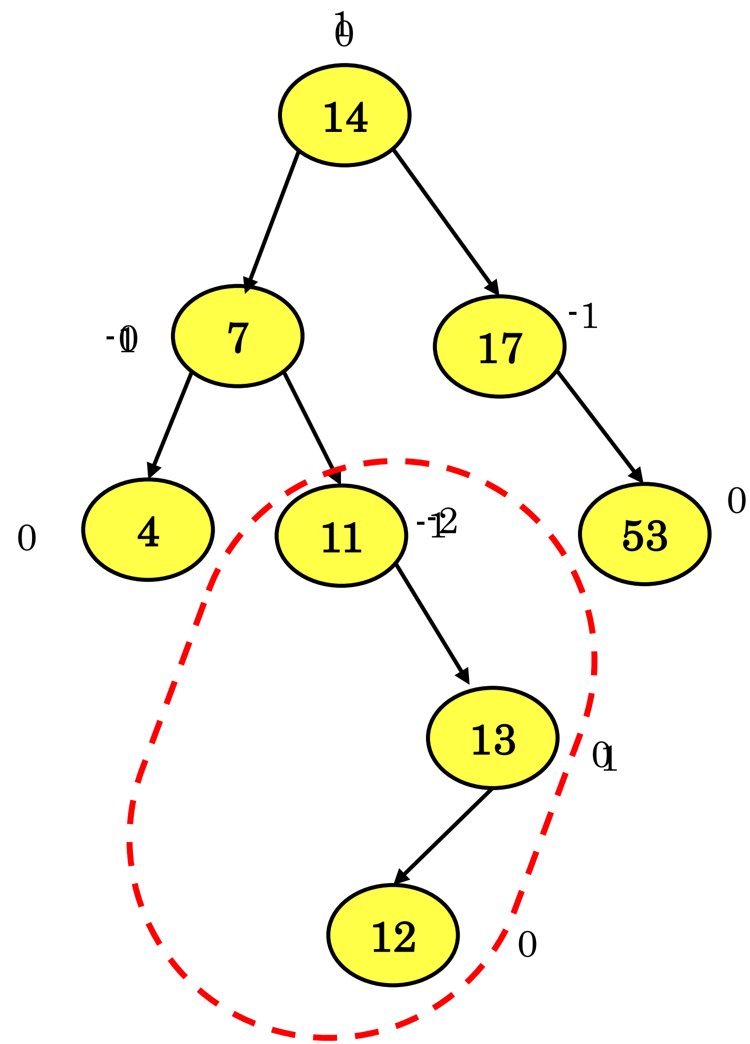
Construct AVL Tree

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



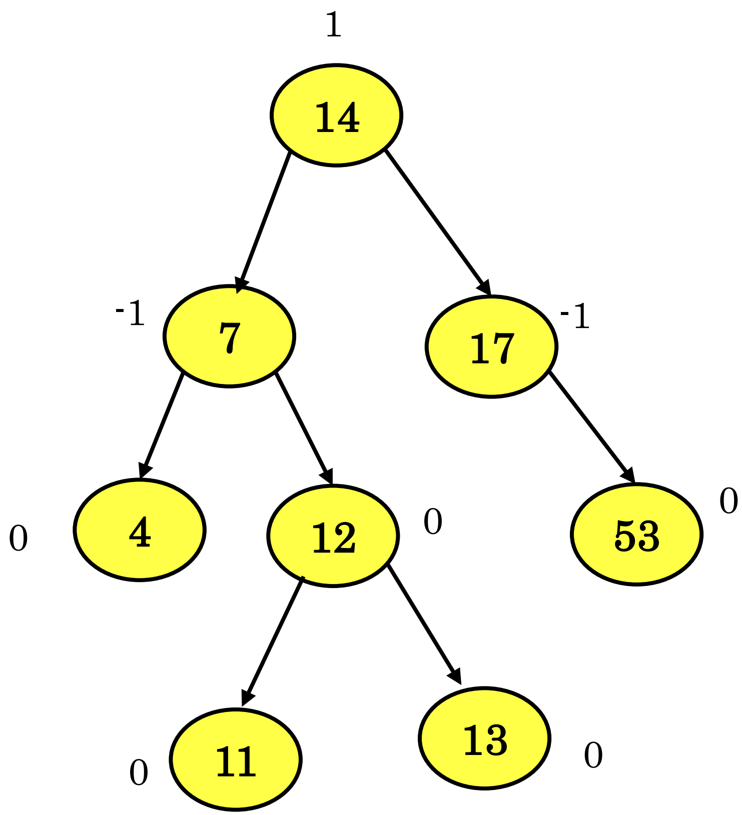
Construct AVL Tree

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



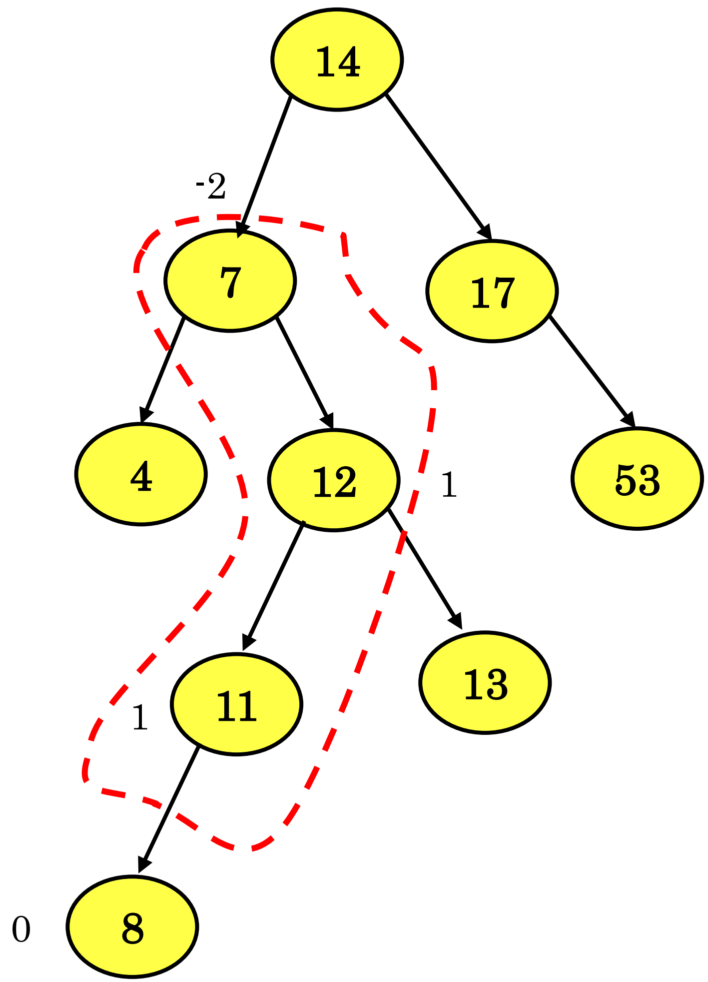
Construct AVL Tree

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



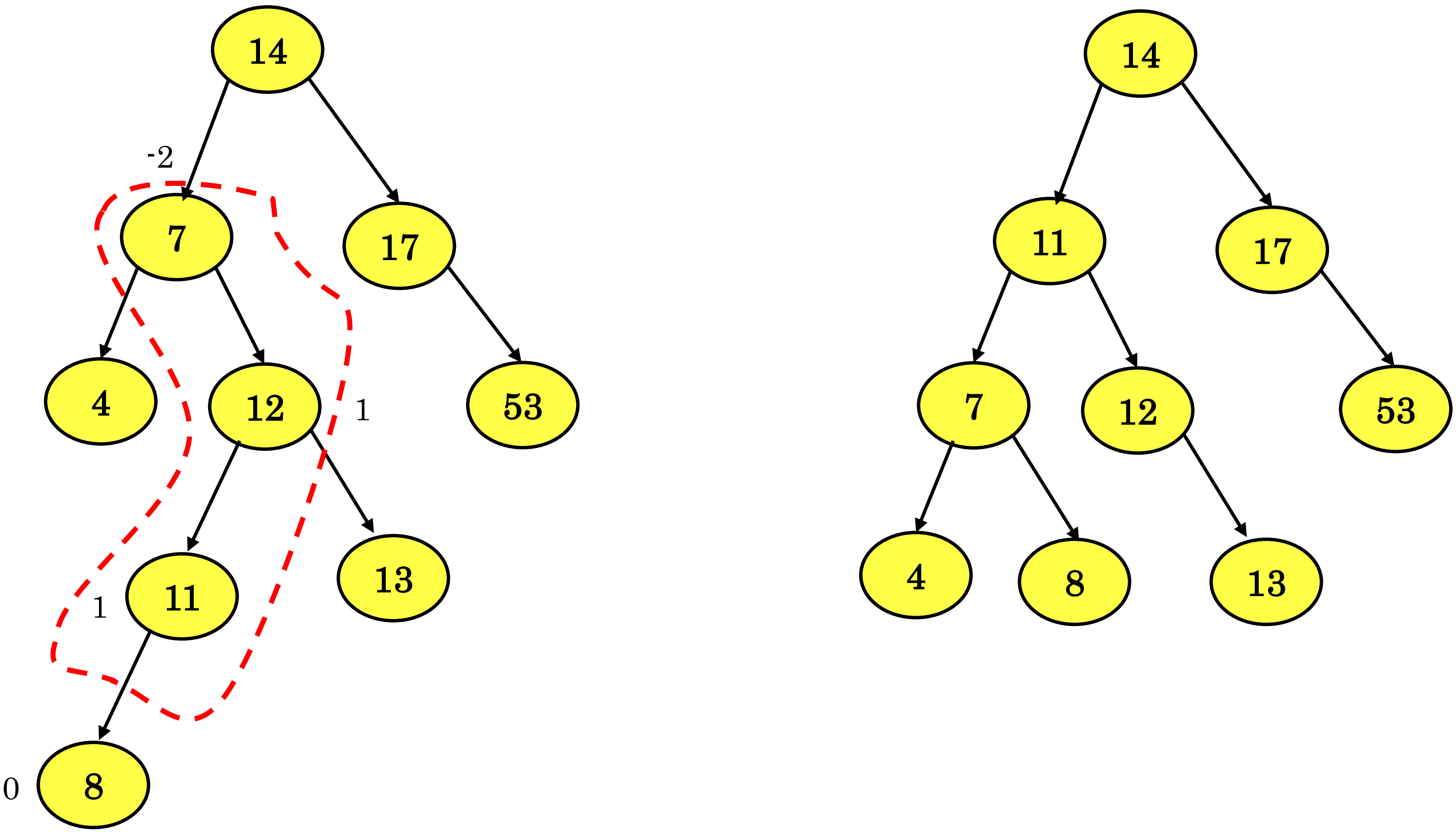
Construct AVL Tree

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



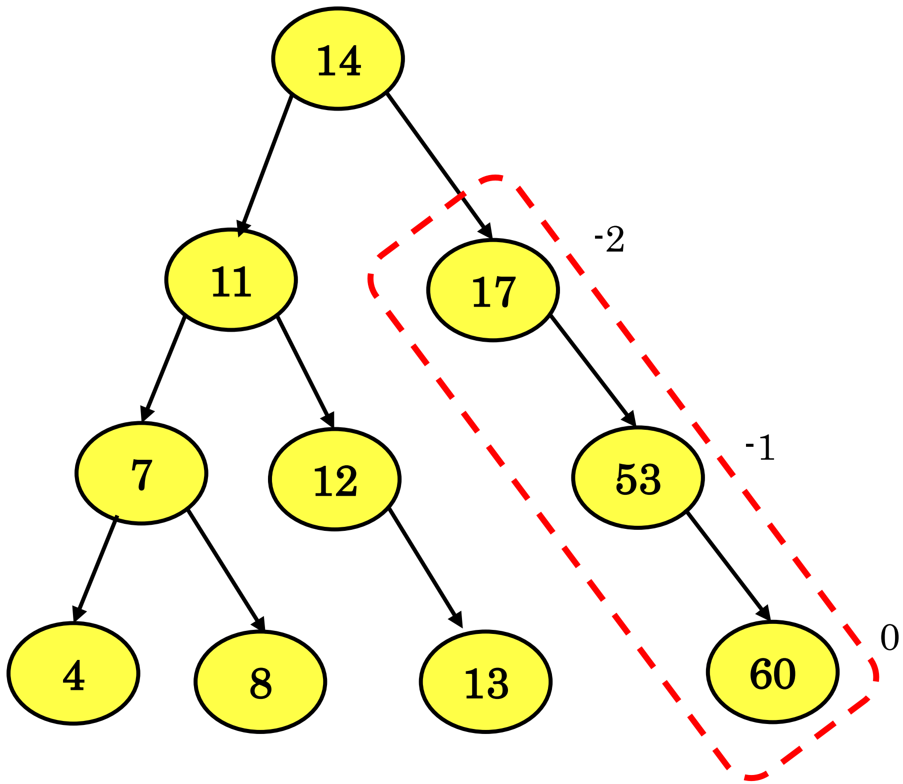
Construct AVL Tree

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



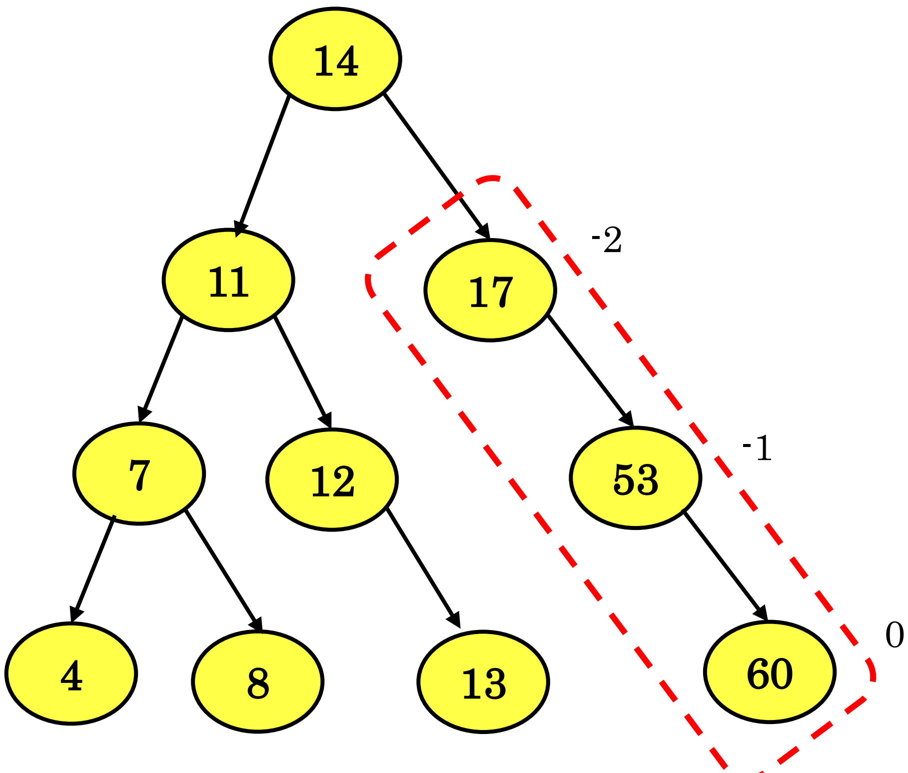
Construct AVL Tree

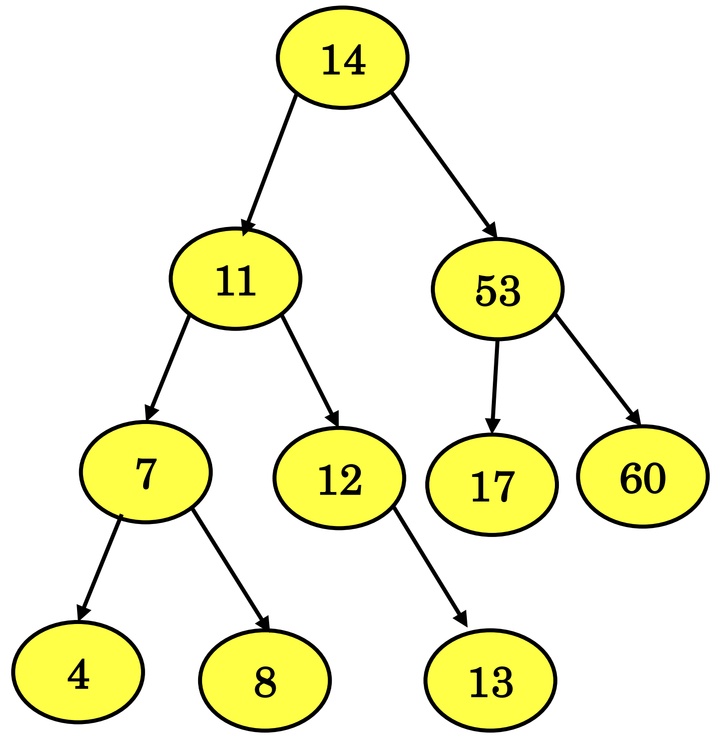
14 17 11 7 53 4 13 12 8 60 19 16 20



Construct AVL Tree

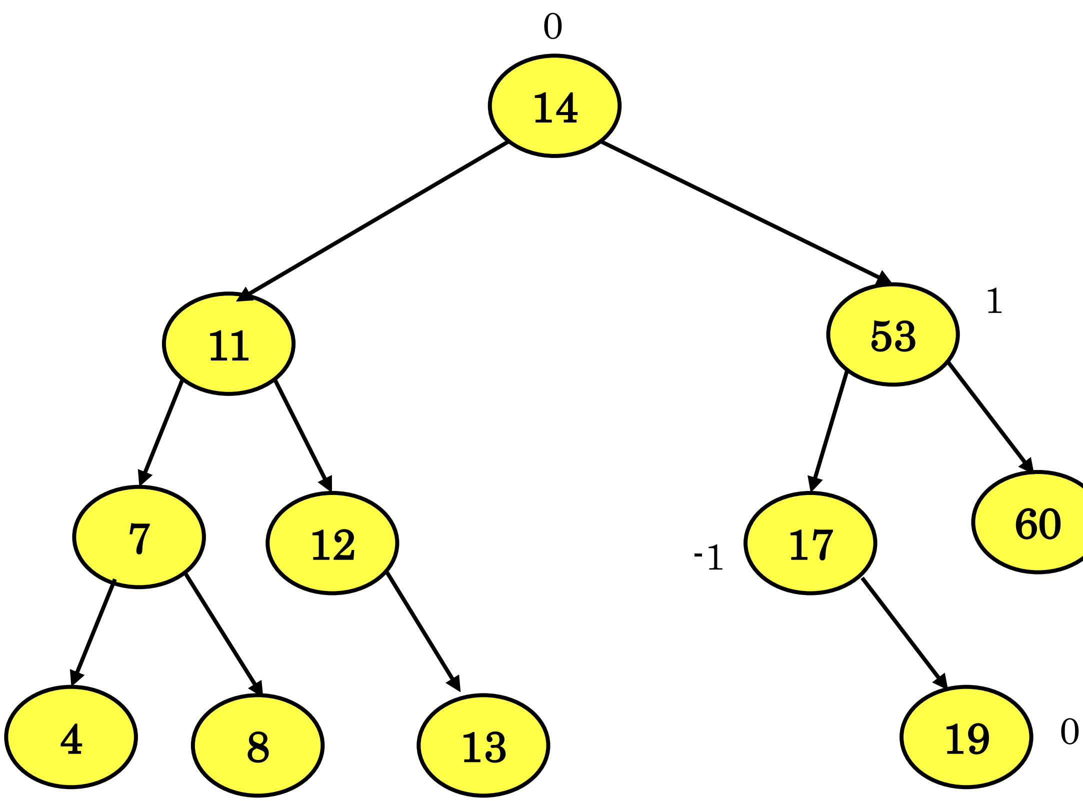
14 17 11 7 53 4 13 12 8 60 19 16 20





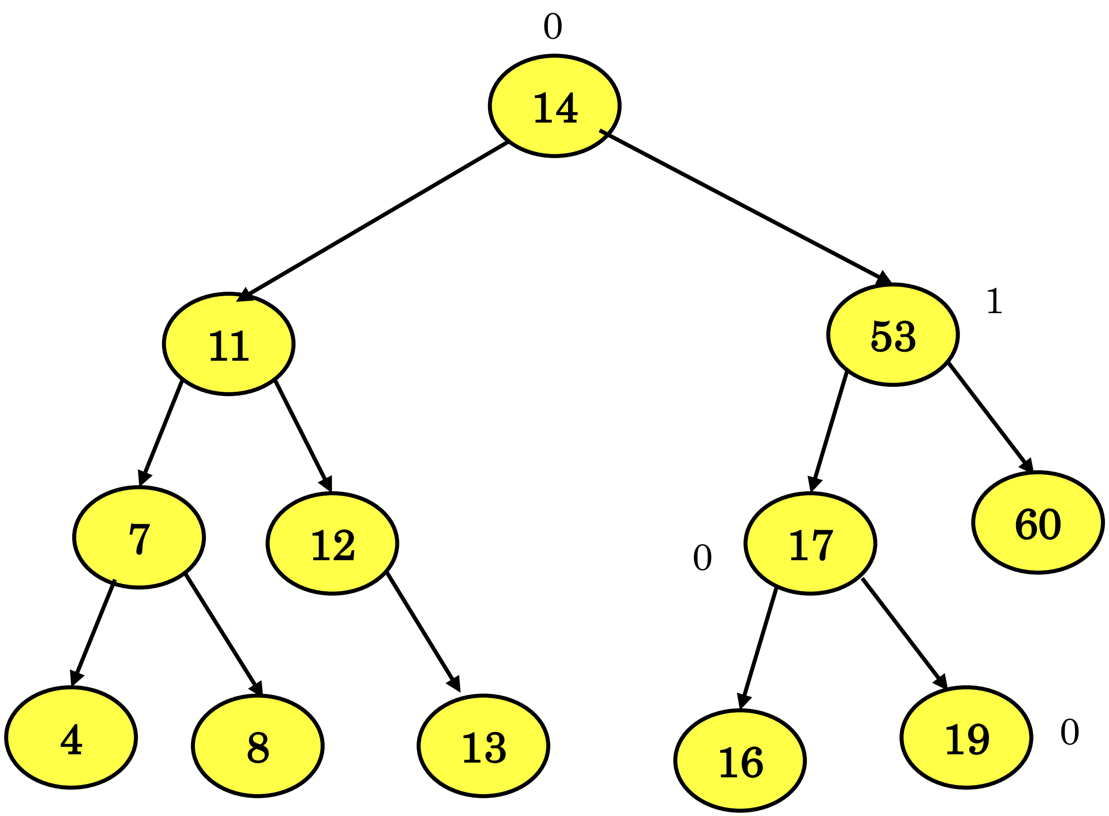
Construct AVL Tree

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



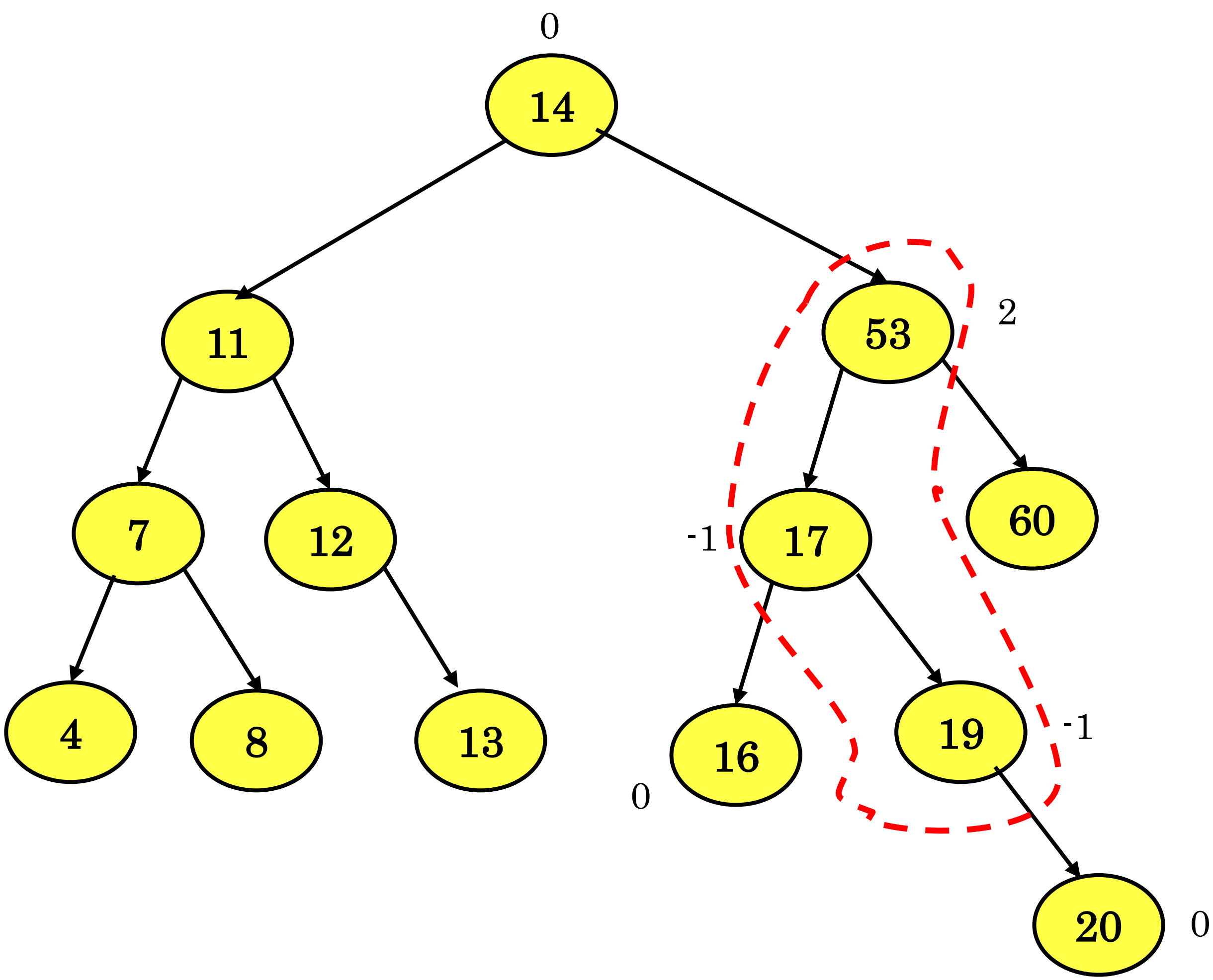
Construct AVL Tree

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



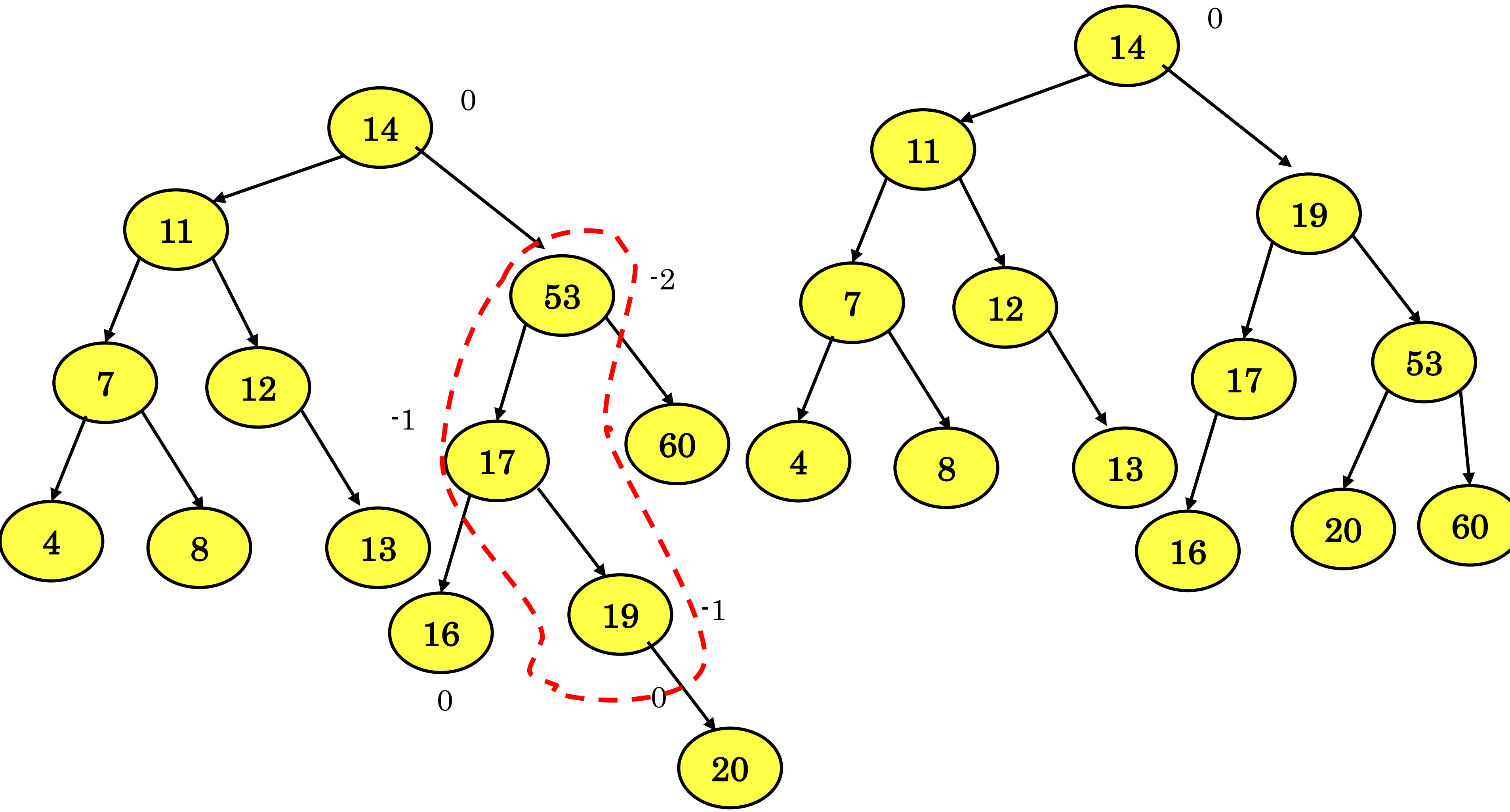
Construct AVL Tree

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



Construct AVL Tree

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



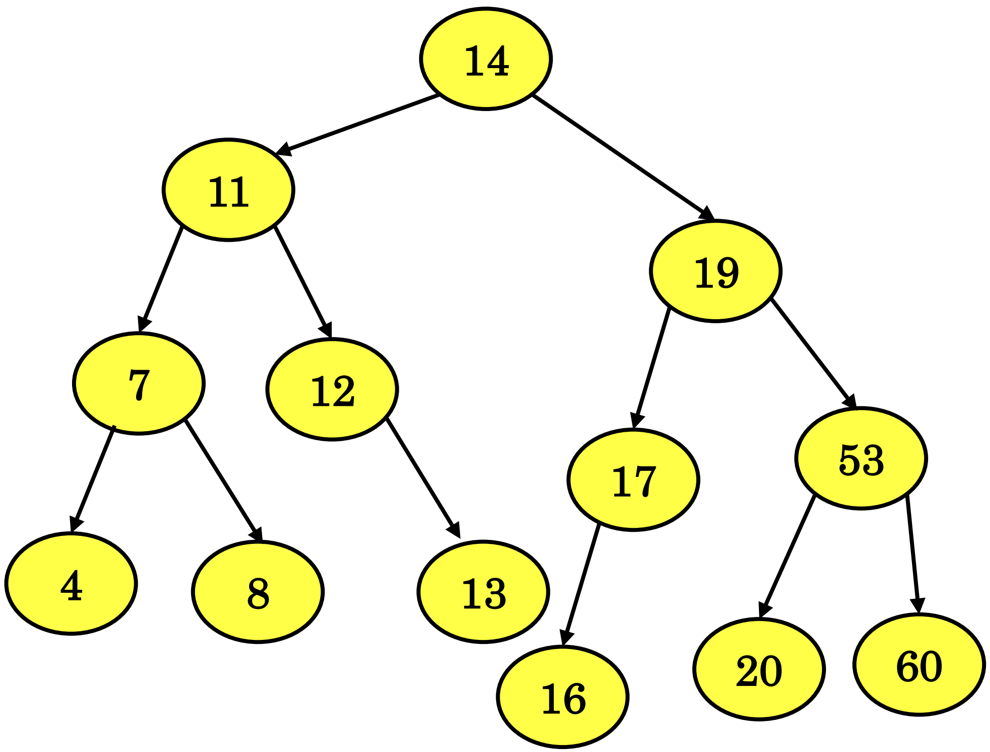
Deletion in AVL Tree

Deletion is same as binary search tree

After every deletion you have to balance the tree

8   7   11   14   17

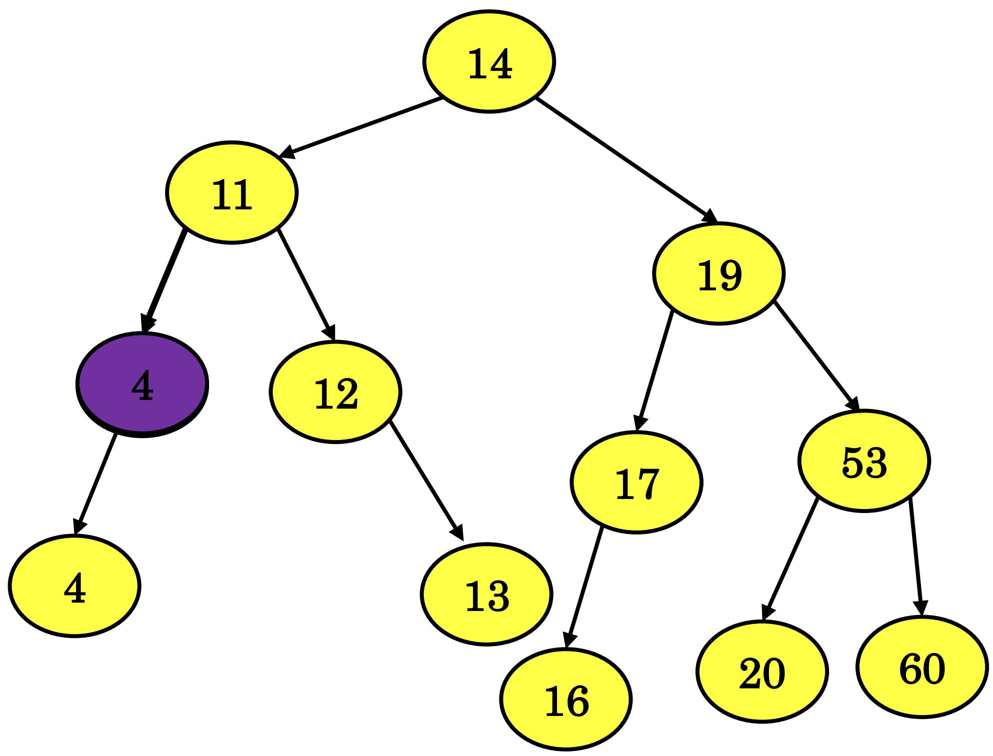
Delete 8



Deletion in AVL Tree

8   7   11   14   17

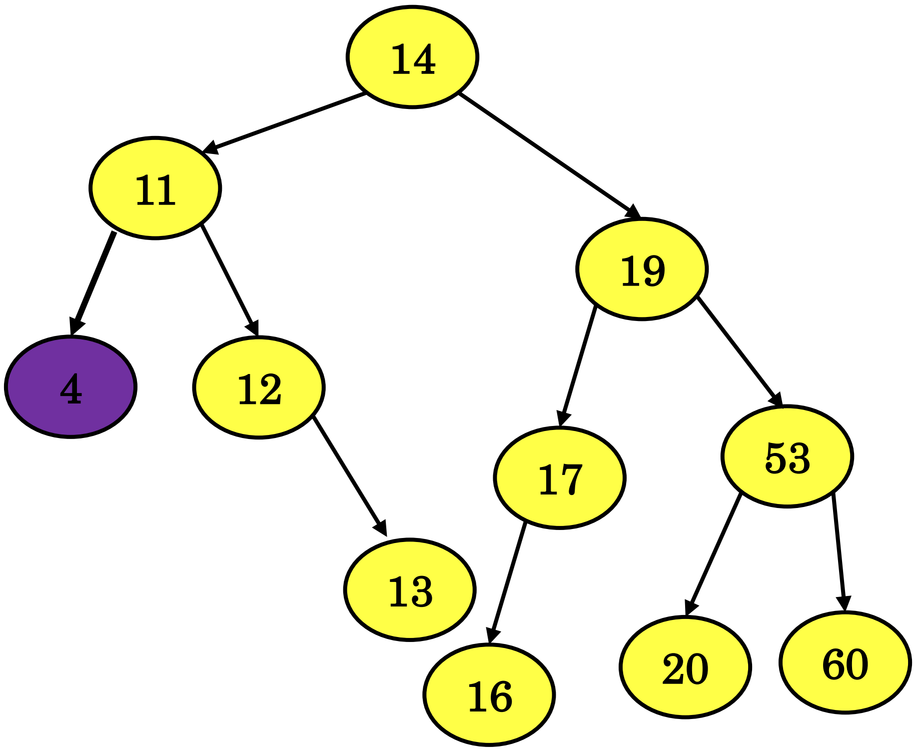
Delete 7



Deletion in AVL Tree

8   7   11   14   17

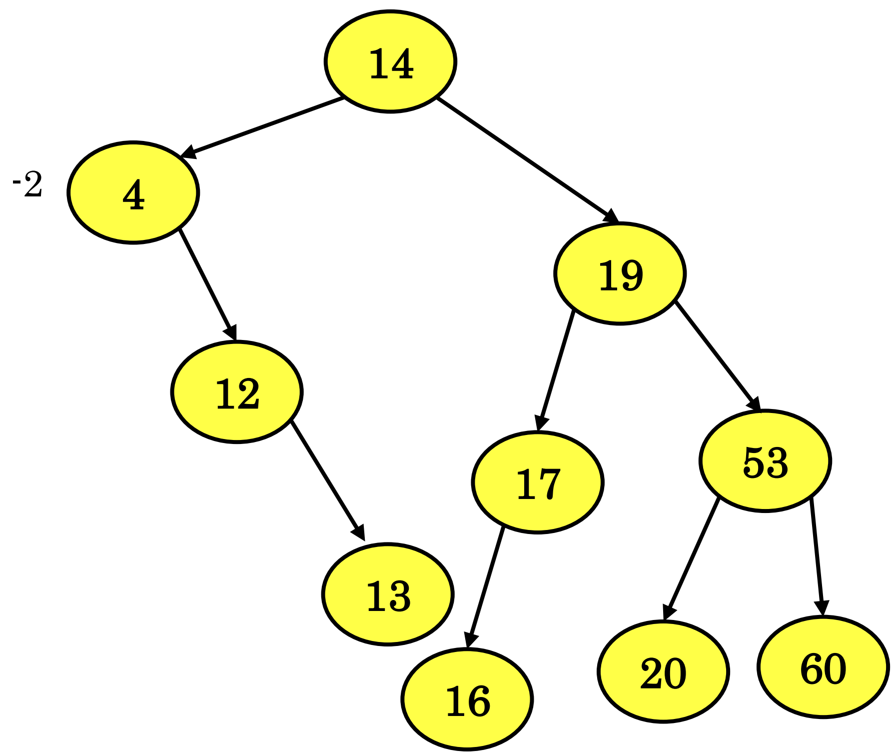
Delete 11



Deletion in AVL Tree

8 7 11 14 17

Delete 11

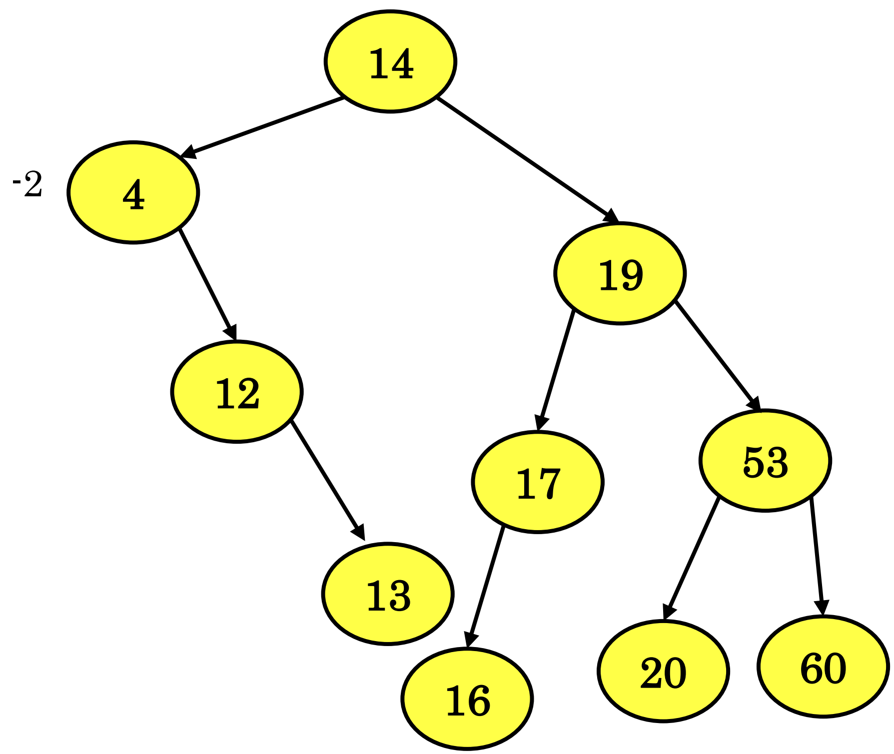


Replace by inorder predecessor or successor

Deletion in AVL Tree

8   7   11   14   17

Delete 11

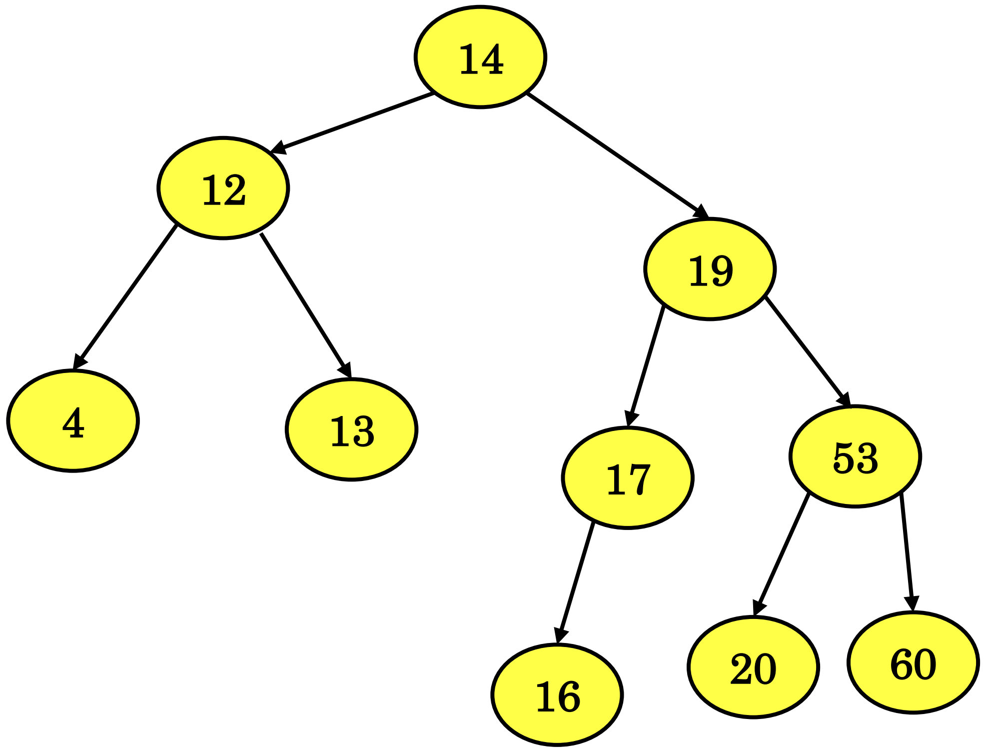


Replace by inorder predecessor or successor

Deletion in AVL Tree

8 7 11 14 17

Balanced

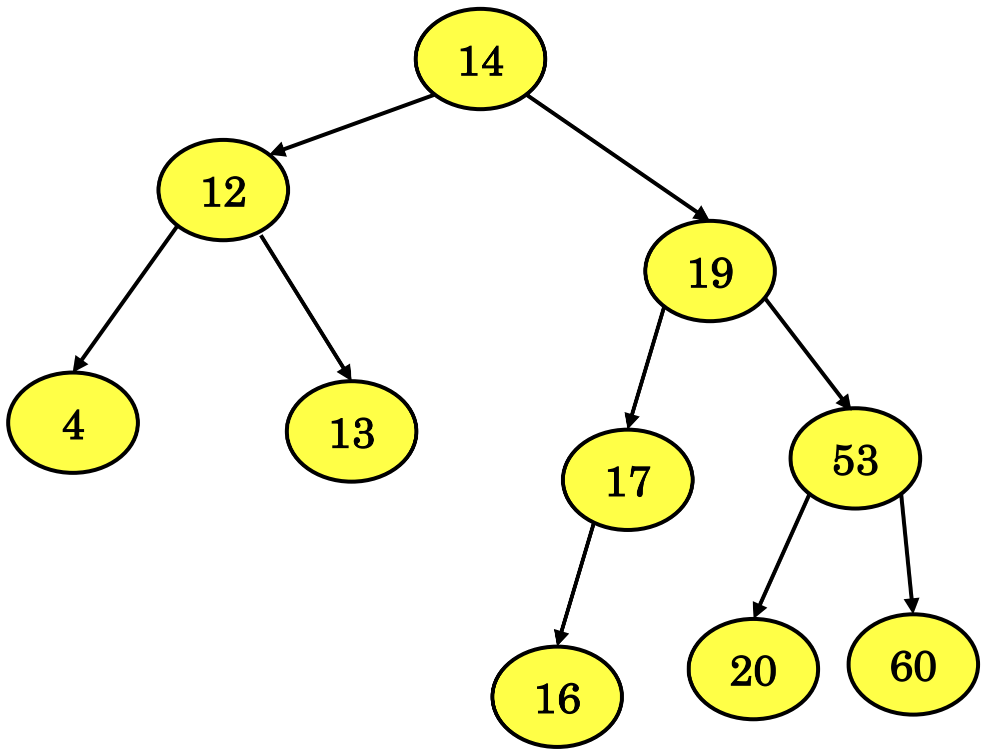


Performed Left Rotation

Deletion in AVL Tree

8 7 11 14 17

Delete 14

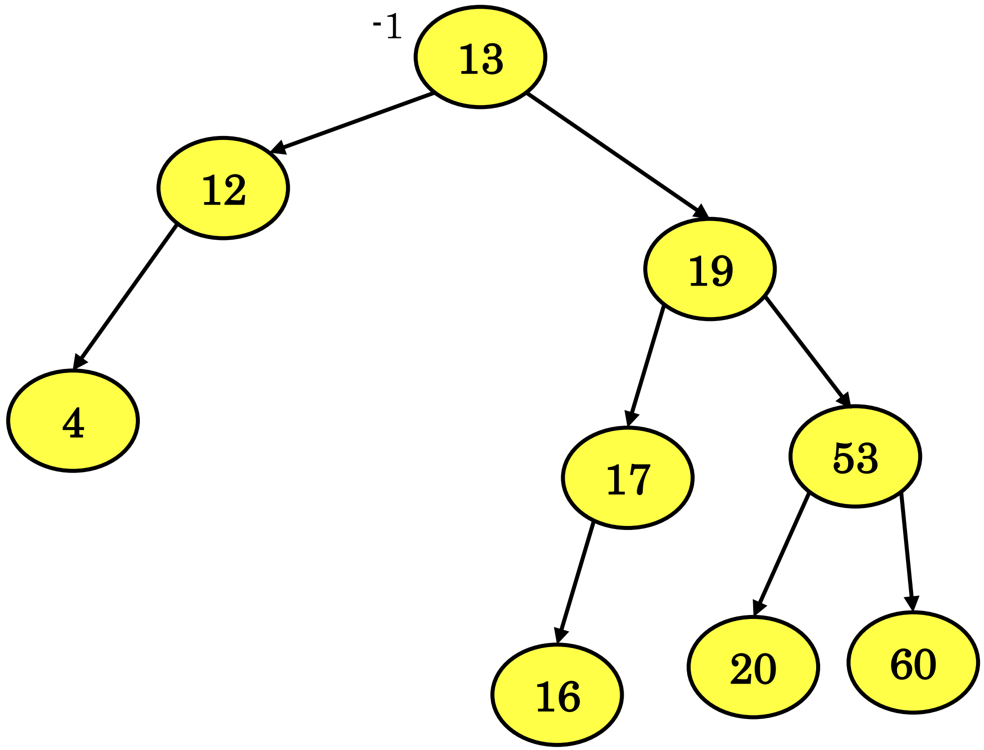


Replace by inorder predecessor or successor

Deletion in AVL Tree

8   7   11   14   17

Delete 14

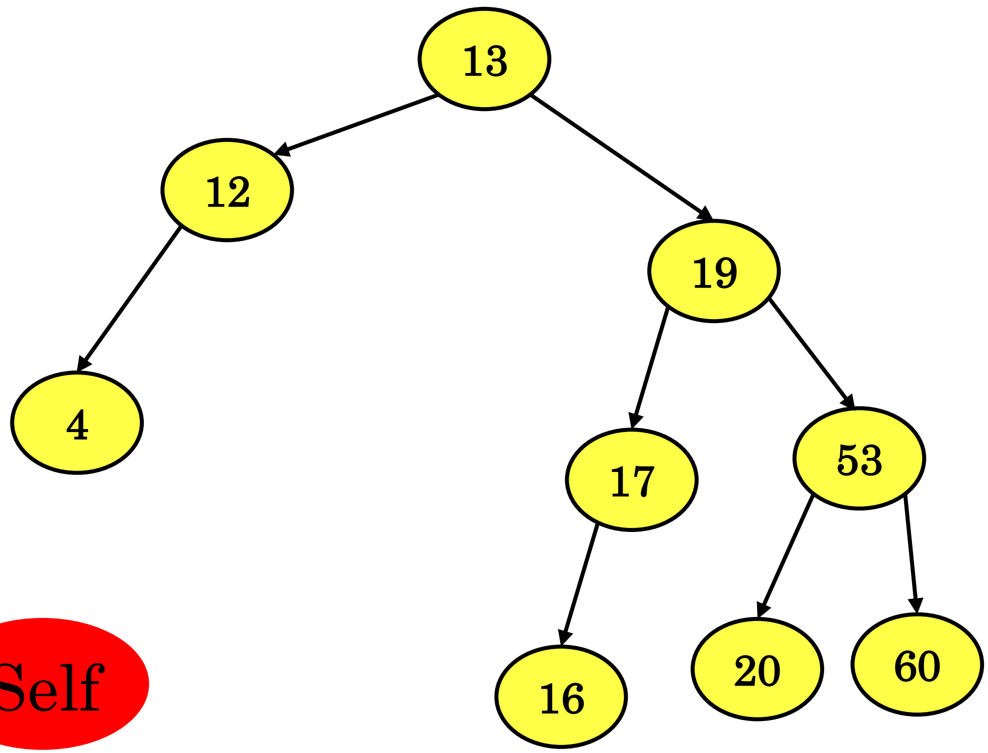


Replace by inorder predecessor or successor

Deletion in AVL Tree

8 7 11 14 17

Delete 17



Self

AVL Tree Complexity

Insertion
Deletion
Searching

Best Case → $O(\log n)$
Avg. Case → $O(\log n)$
Worst Case → $O(\log n)$

Search → Best Case → O(1)

B-Tree

- ✓ Balanced m−way tree
- ✓ A BST in which a node can have more than one key and more than 2 children
- ✓ Maintains stored data
- ✓ All leaf node must be at same level
- ✓ B−tree of order m has following properties
- ✓ Every node has maximum m children
- ✓ Min Children →
  - ✓ Leaf → 0
  - ✓ Root → 2
  - ✓ Internal nodes $= \frac{m}{2} \rceil$

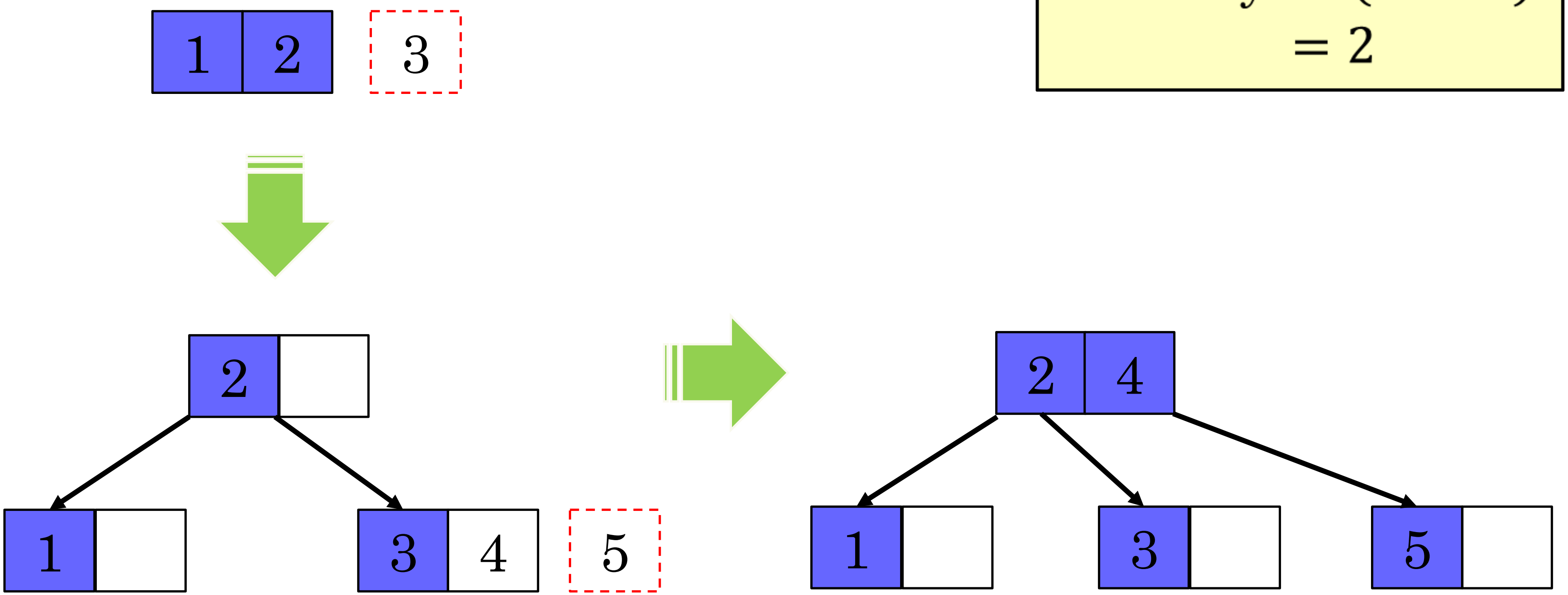
B-Tree

- ✓ Every node has maximum $(m - 1)$ keys
- ✓ Min Keys :
  - ✓ Root Node → 1
  - ✓ All other nodes → $\frac{m}{2} - 1 \rceil$

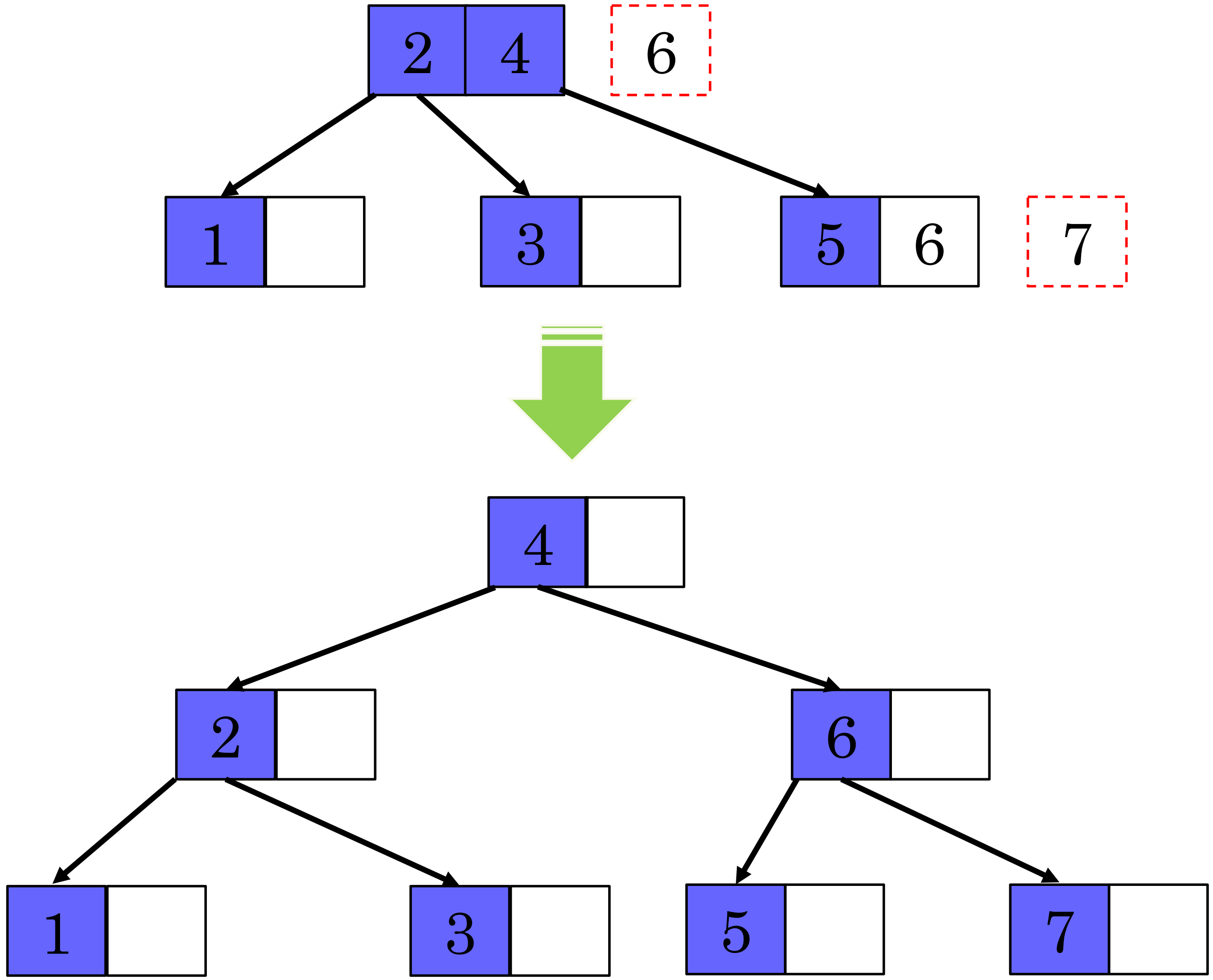
Insertion in B-Tree

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

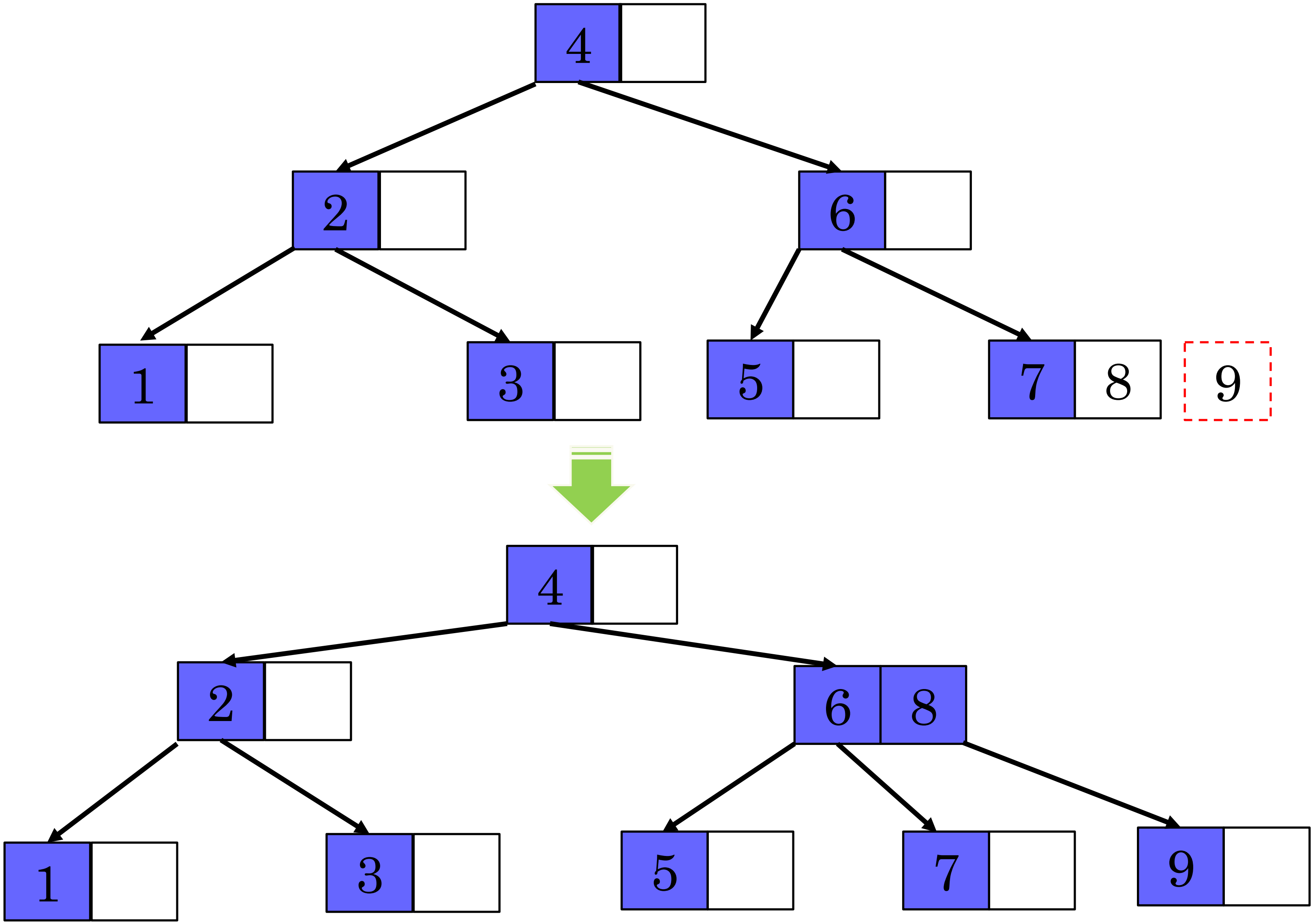
Max key $= (m-1)$
$= 2$



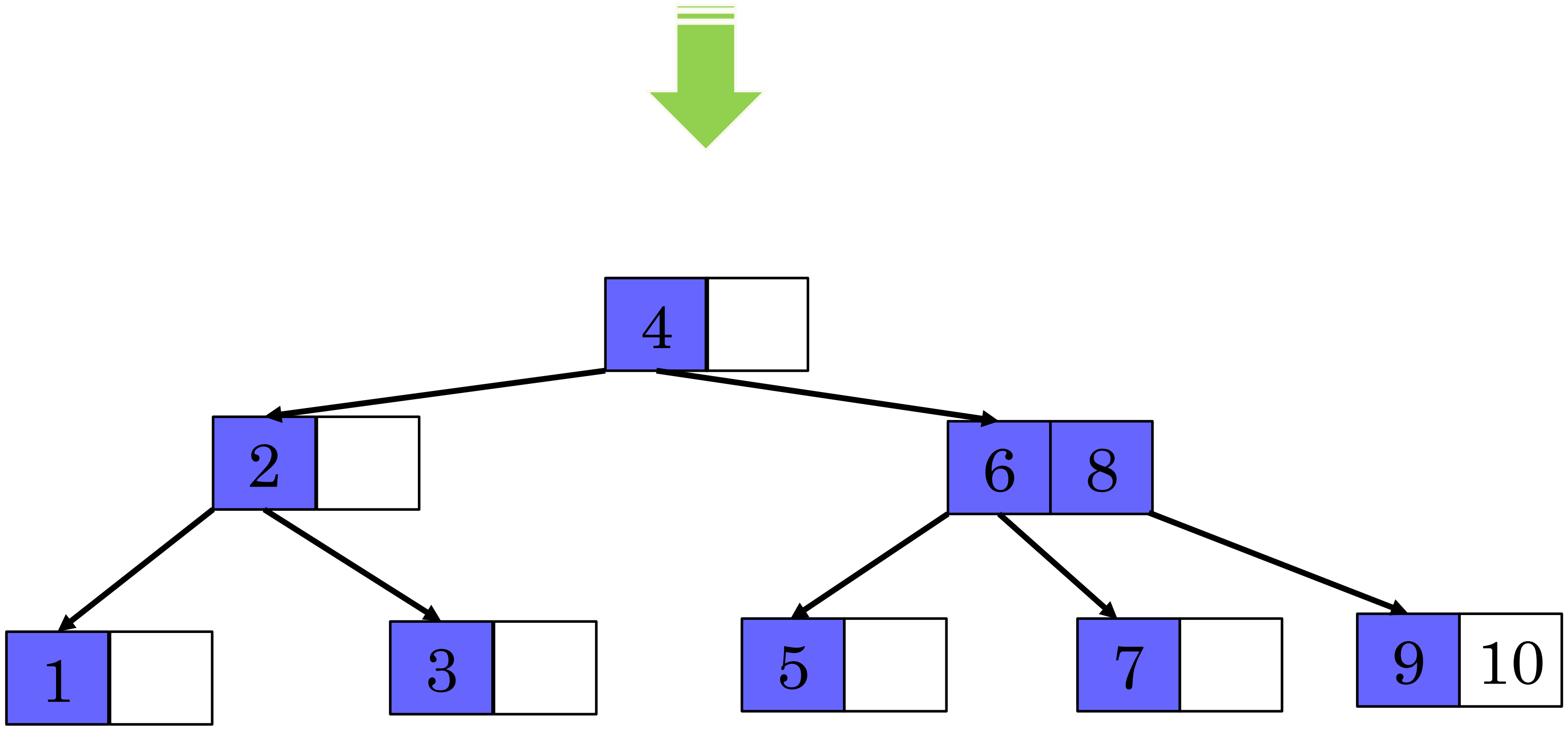
Insertion in B-Tree



Insertion in B-Tree



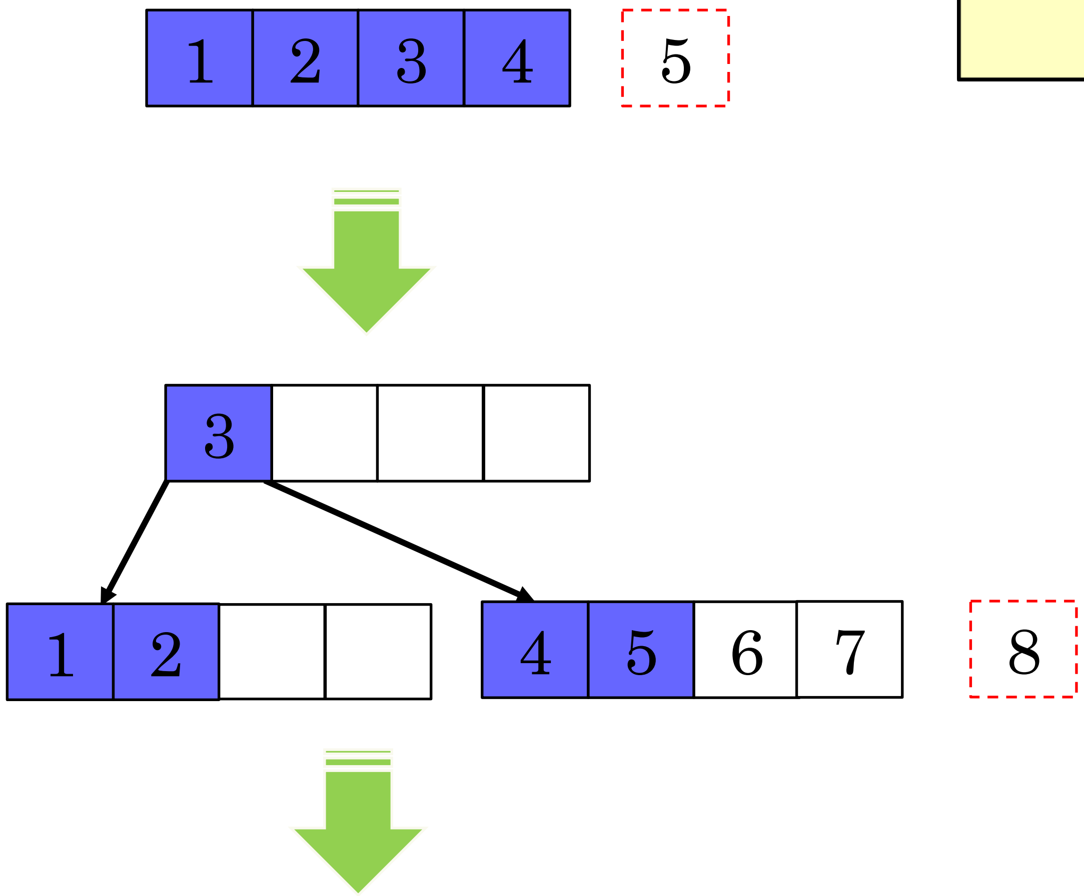
Insertion in B-Tree



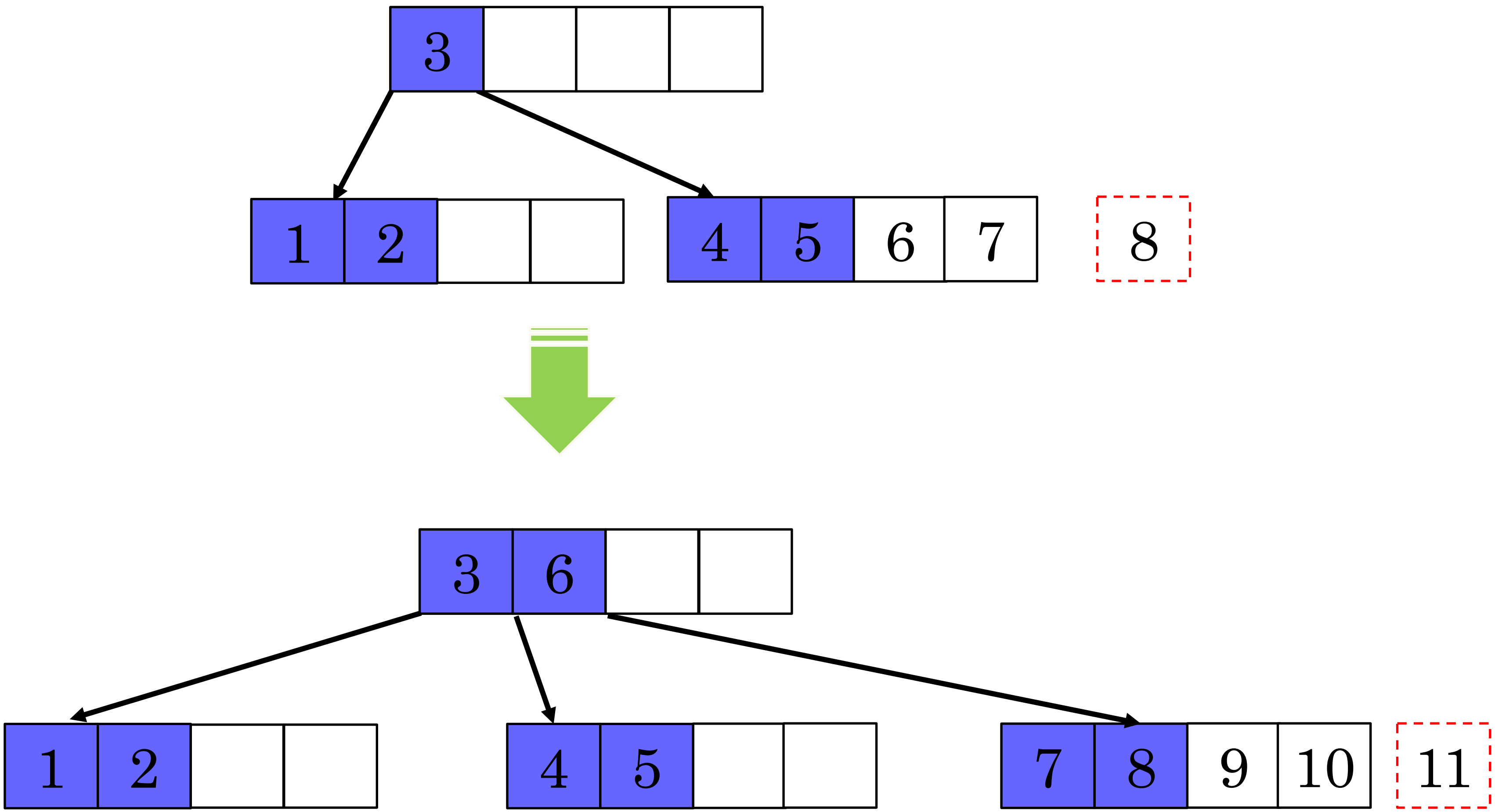
Insertion in B-Tree

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

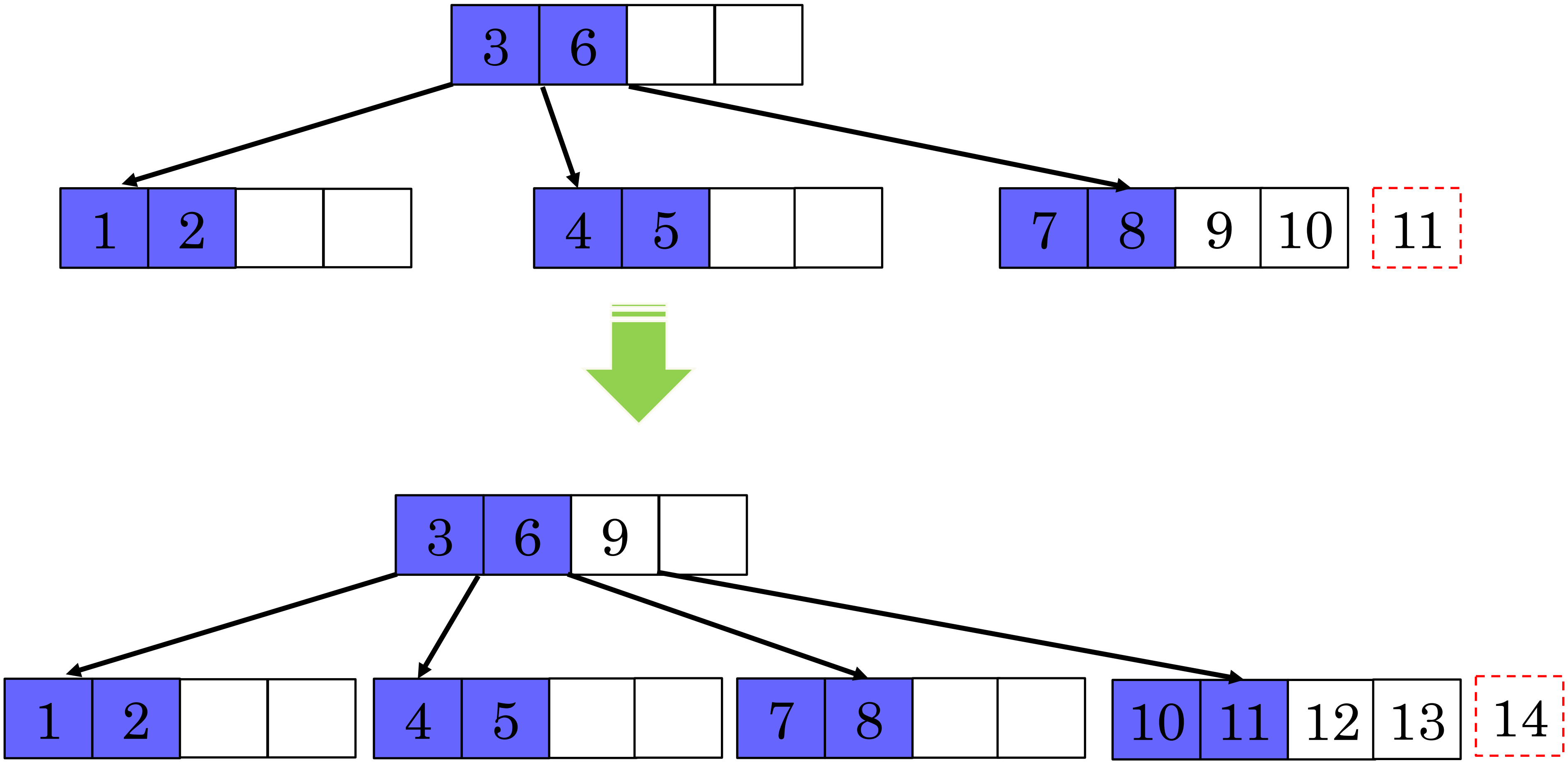
Max key $= m - 1$
$= 4$



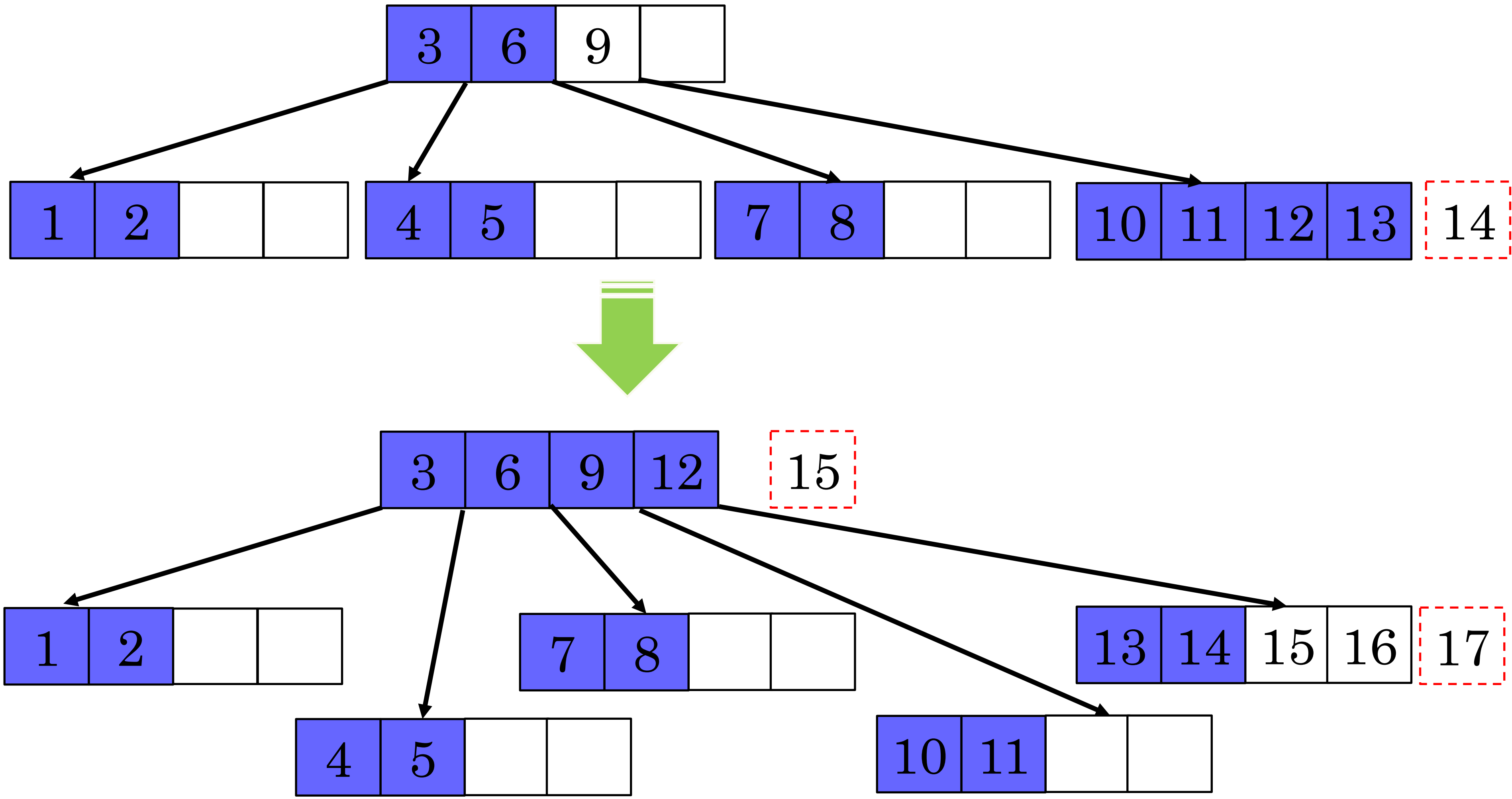
Insertion in B-Tree



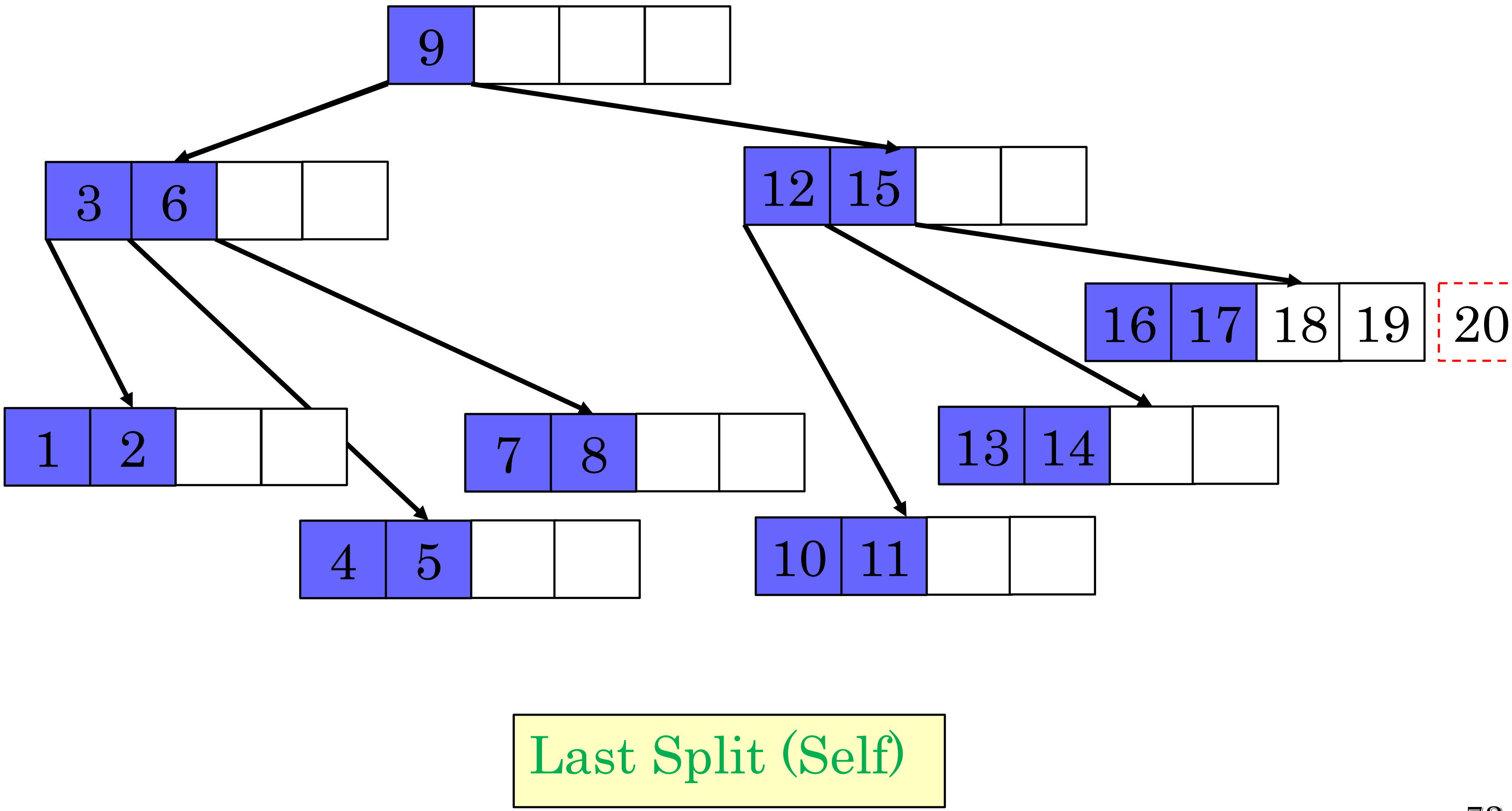
Insertion in B-Tree



Insertion in B-Tree



Insertion in B-Tree

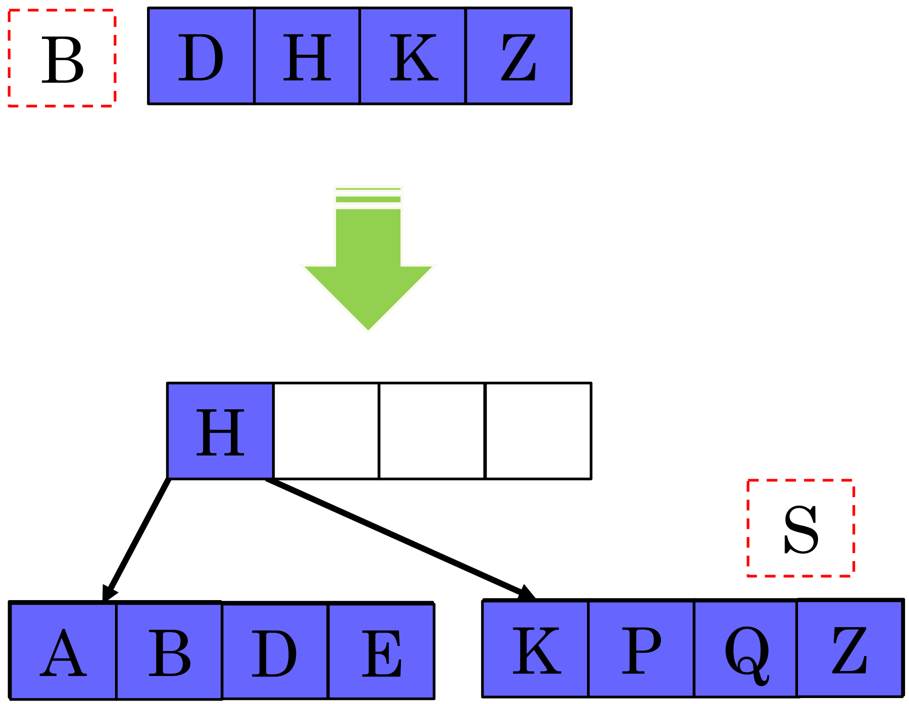


Insertion in B-Tree

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

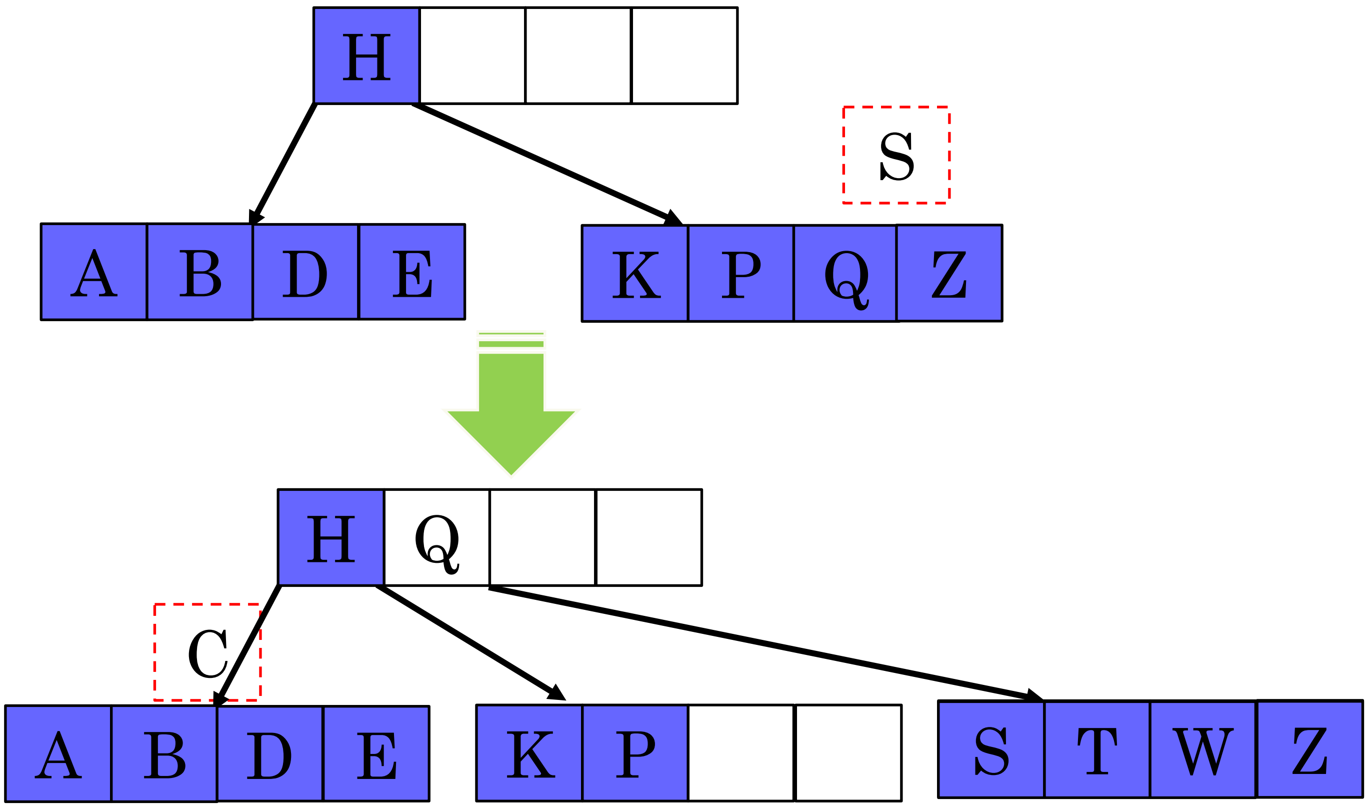


Insertion in B-Tree

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

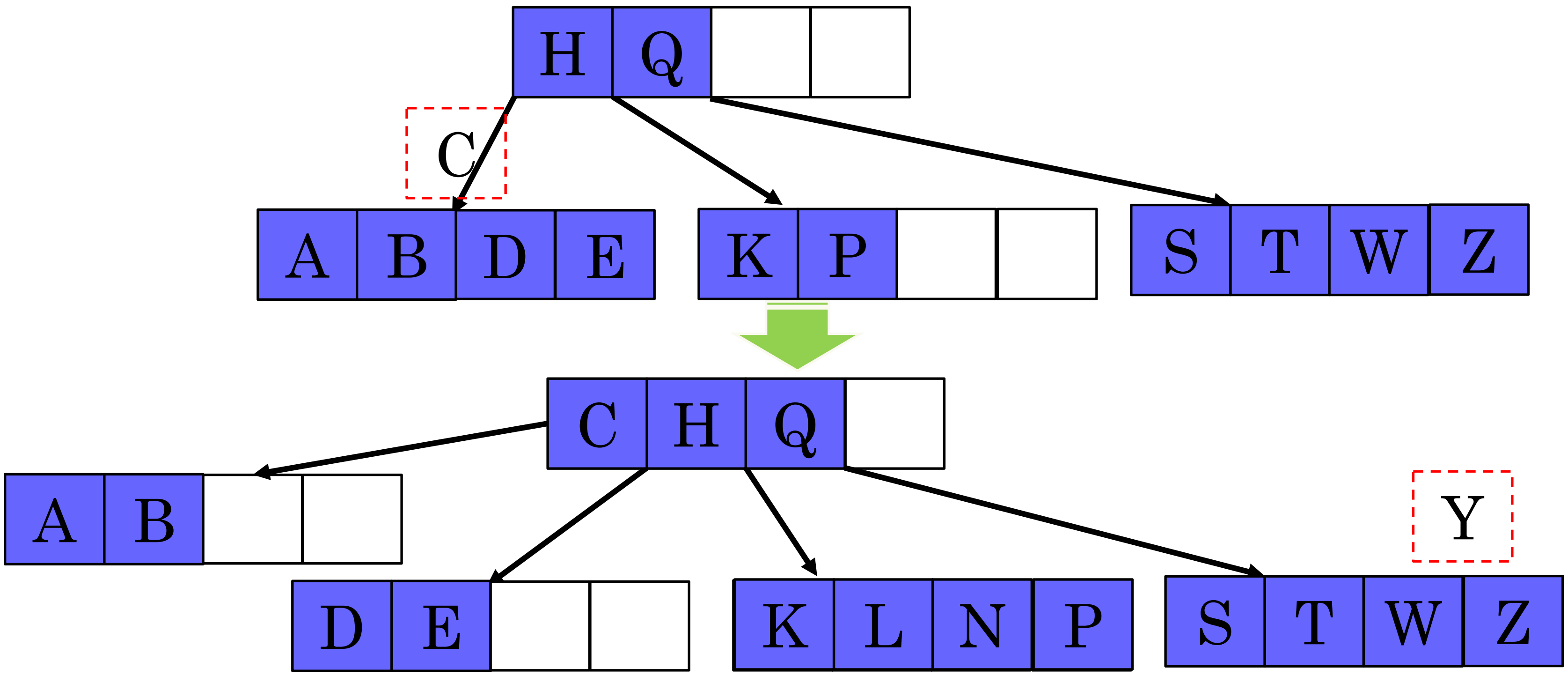


Insertion in B-Tree

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

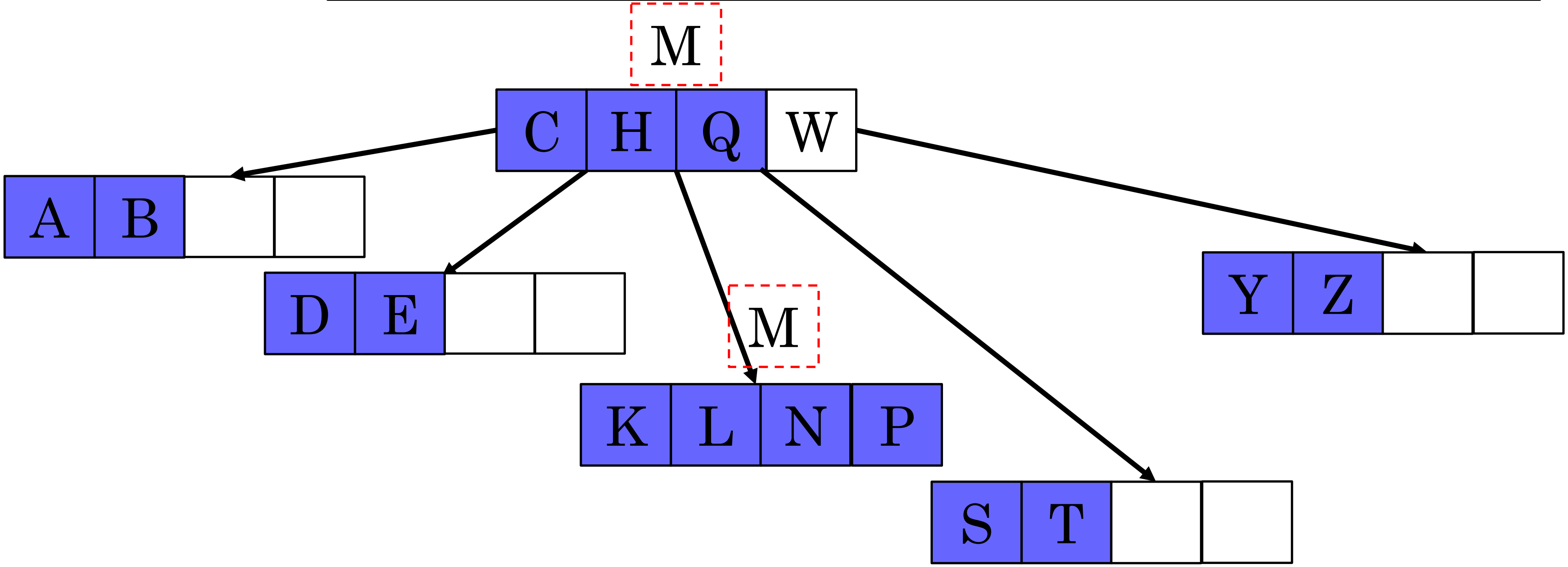


Insertion in B-Tree

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

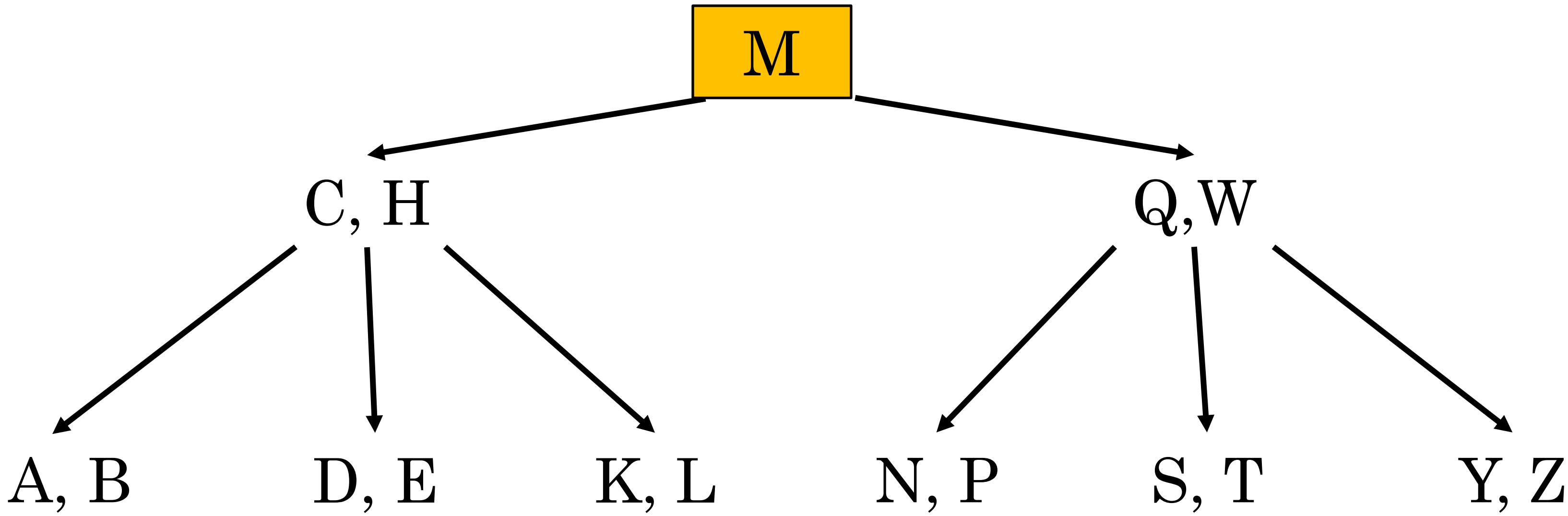


Insertion in B-Tree

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



THANK YOU