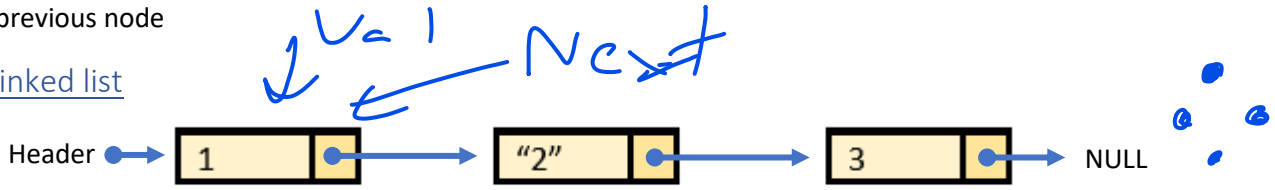
Forward & Backward linked list

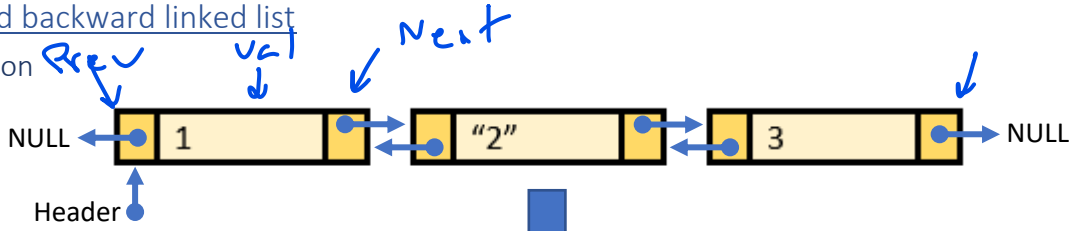
Normal linked list contains only next value but with forward and backward linked list even the last node can trace the previous node

Normal linked list

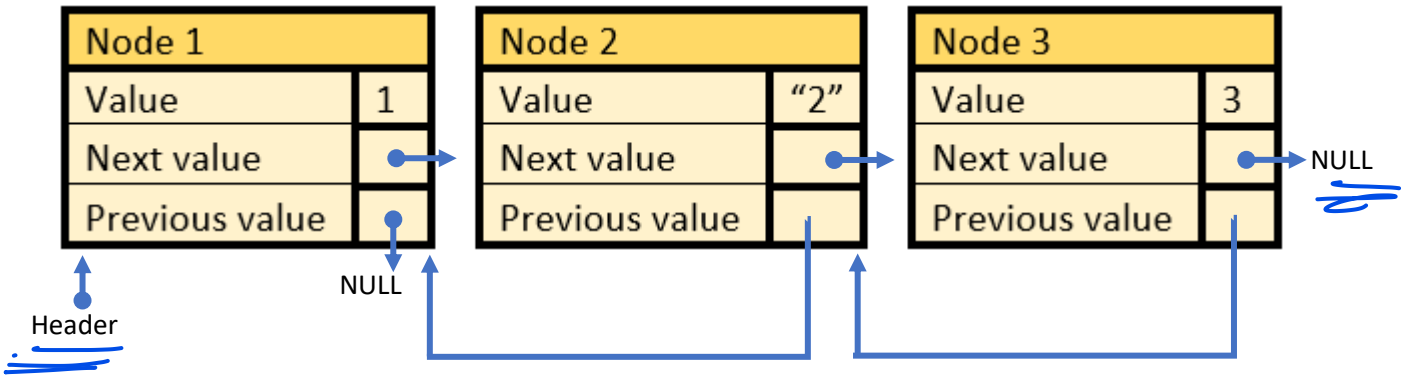


Forward and backward linked list

Normal version

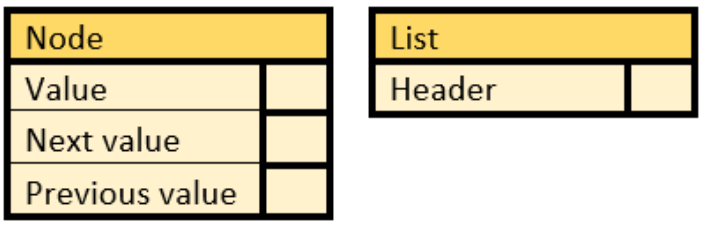


Explains in longer version



Class structure of forward and backward linked list

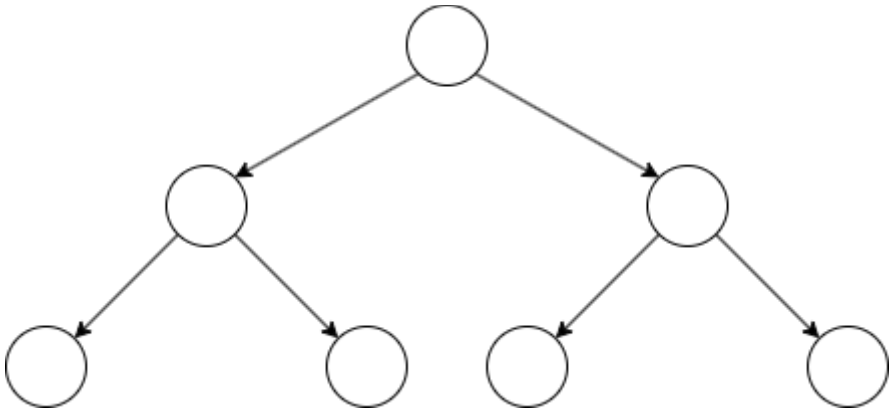
Node	List
class Node(): Value = "" NextVal = "" PrevVal = ""	class List(): Header = ""



Binary tree

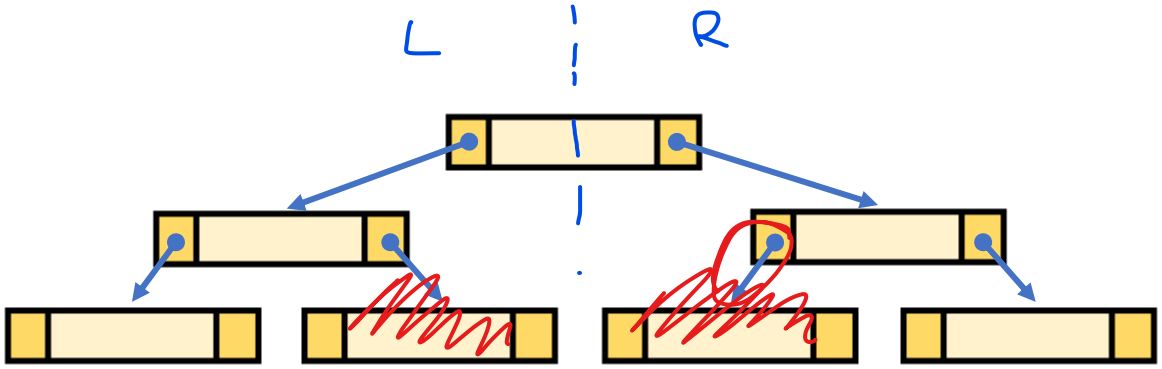
Structure of binary tree

Meaning of “bi” is two. So binary tree is the tree that each node will not contains next value more than two nodes.



Structure of binary tree in normal view

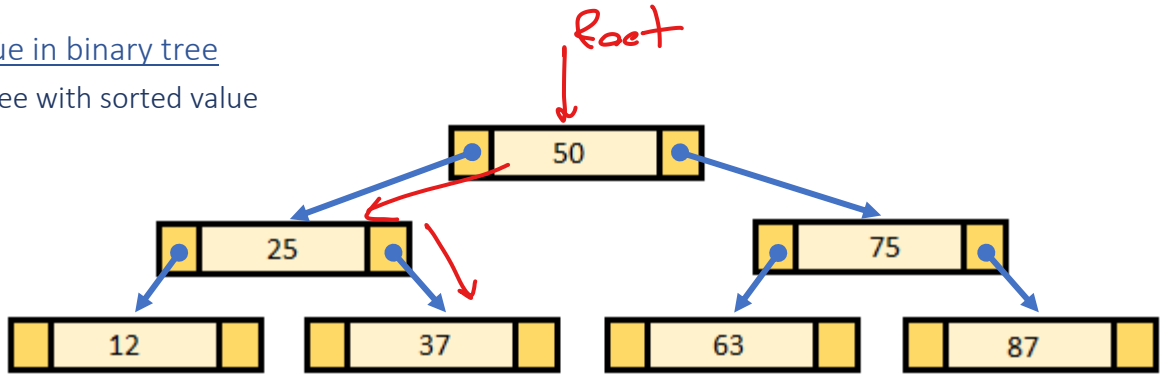
Structure of binary tree with linked list



Structure of binary tree in forward and backward linked list view

Add value in binary tree

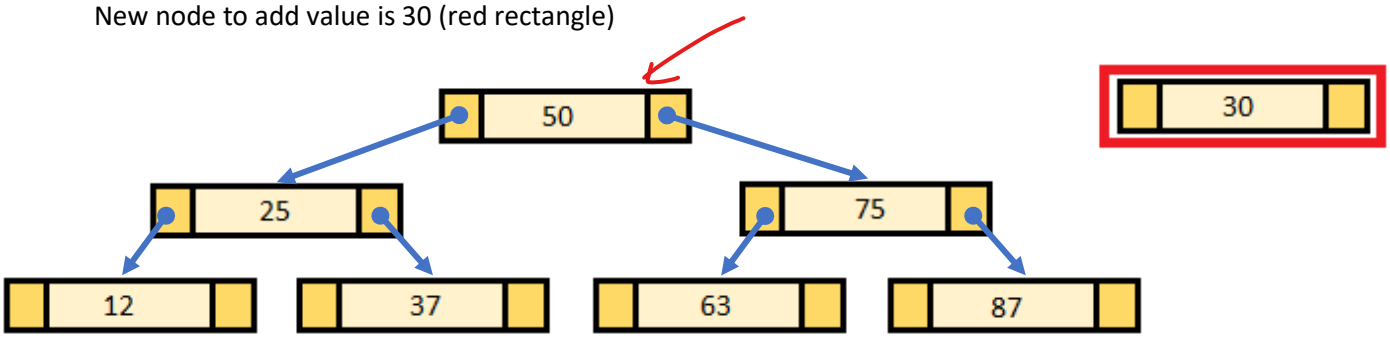
Binary tree with sorted value



Binary tree with value

Adding new node (value = 30)

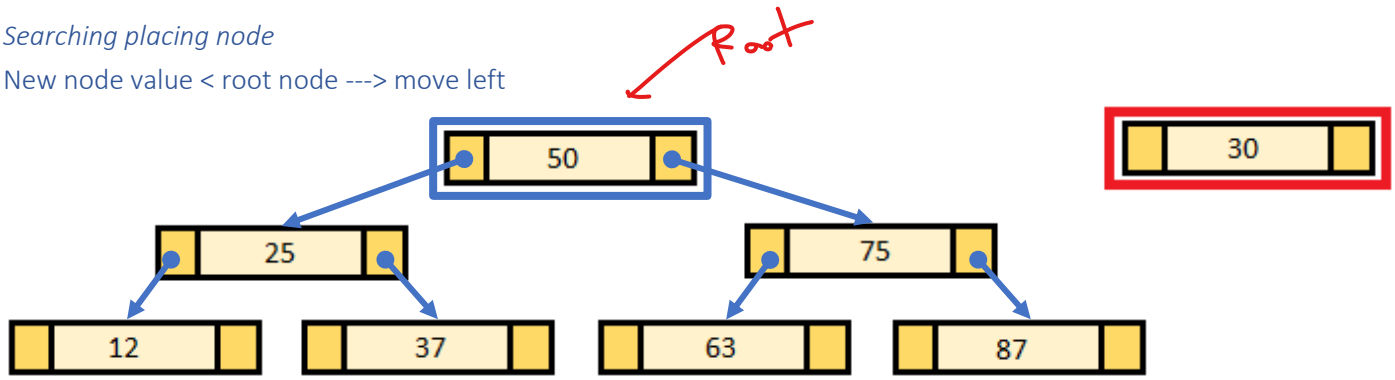
New node to add value is 30 (red rectangle)



Adding new value

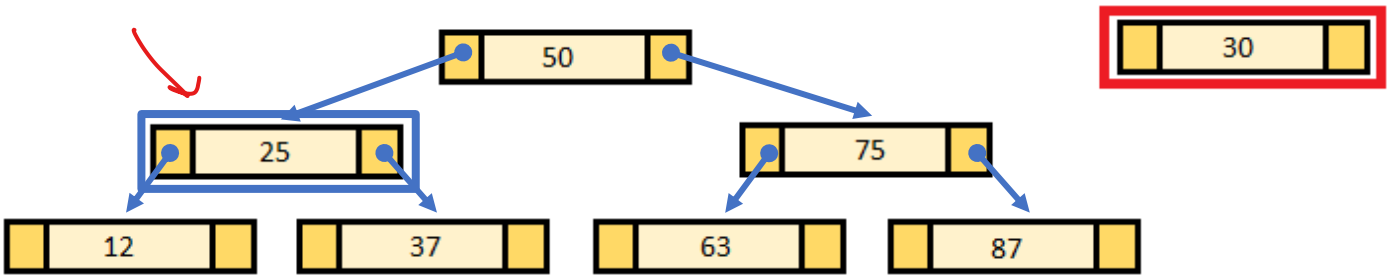
Searching placing node

New node value < root node ---> move left



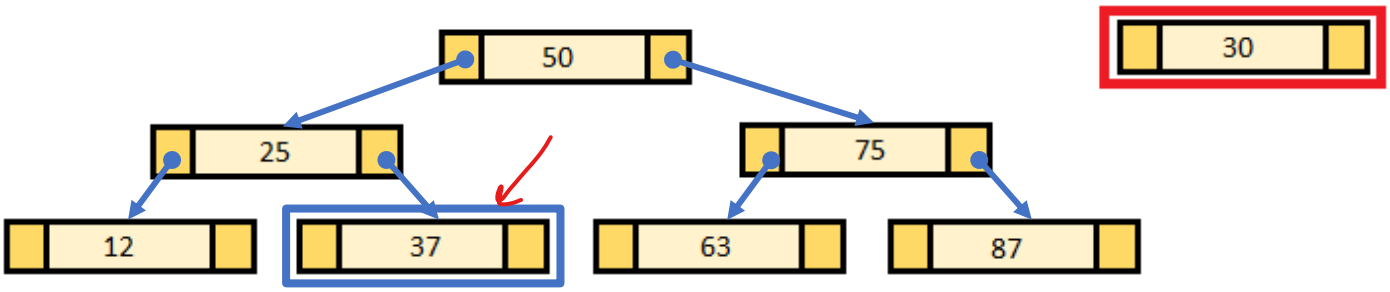
Blue rectangle is current comparison node (value = 50)

New node value > current node ---> move right



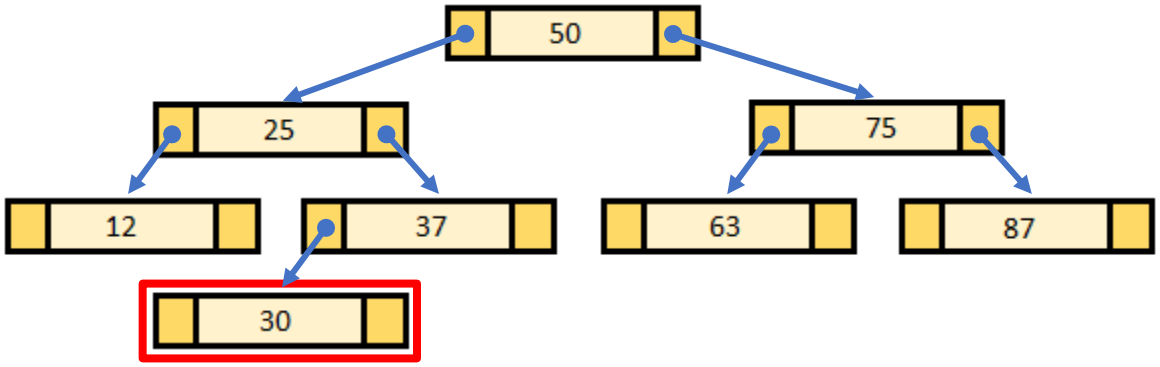
Blue rectangle is current comparison node (value = 25)

New node value < current node ---> move left but there is no further node



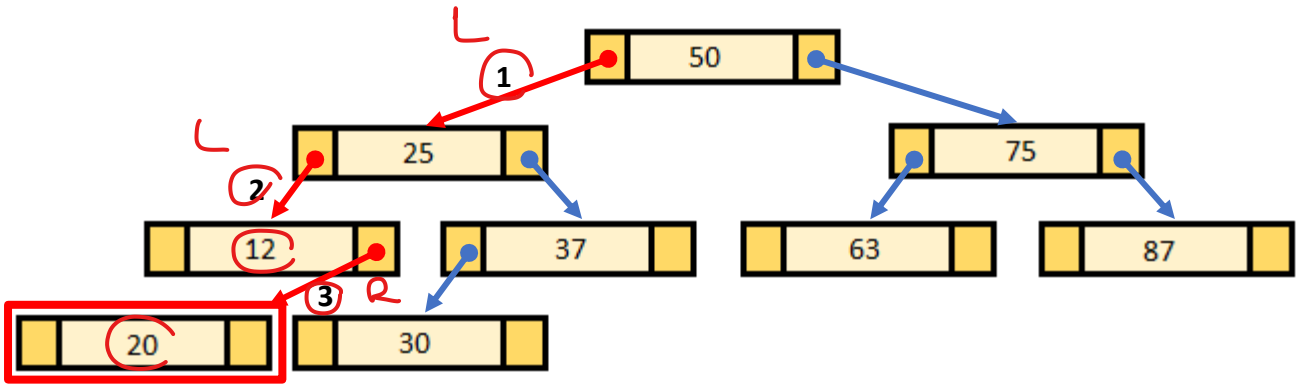
Blue rectangle is current comparison node (value = 37)

Connect the new node



Binary tree after added a new node (red rectangle, value = 30)

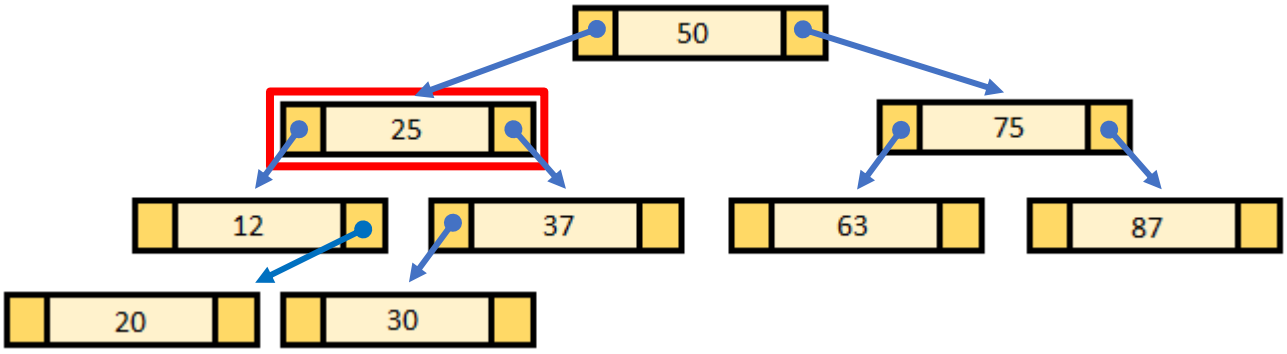
Another example of adding node in binary tree



Binary tree after added a new node (red rectangle, value = 20)

Delete node in binary tree

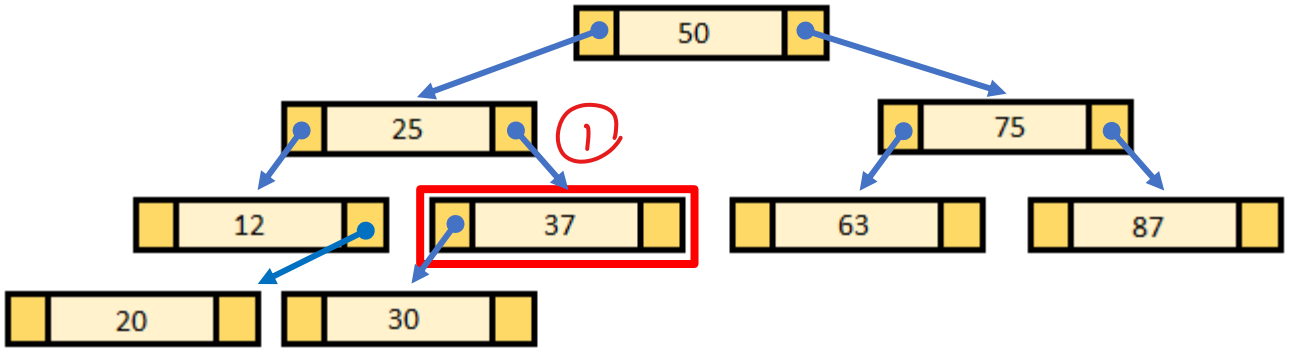
Looking for a node to delete (red rectangle, value = 25)



Node that needed to delete (red rectangle, value = 25)

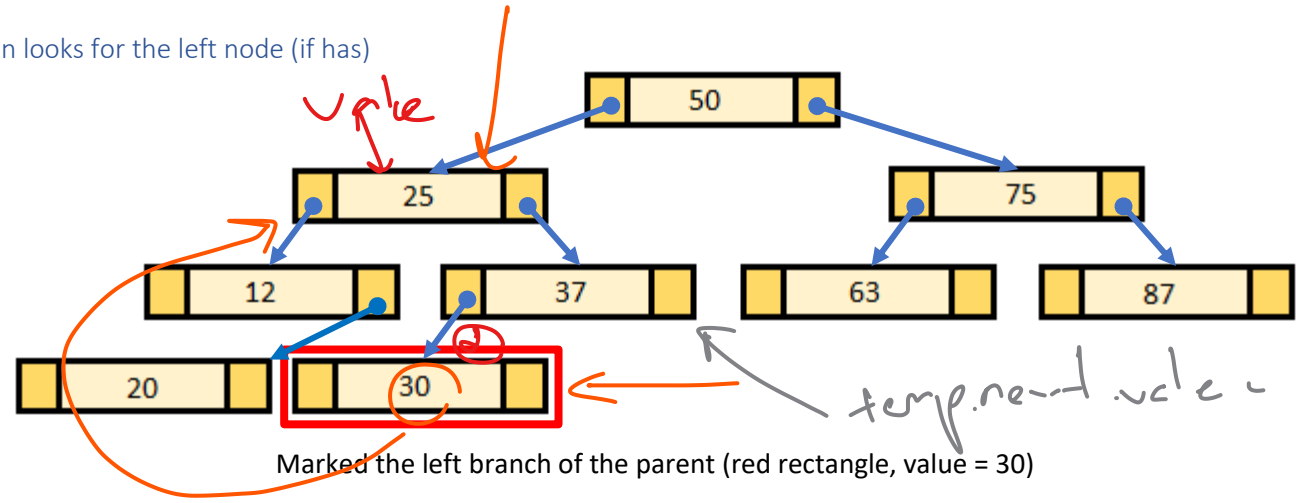
Looking for node to replace

Look for the right of the node that needed to delete first (if has)



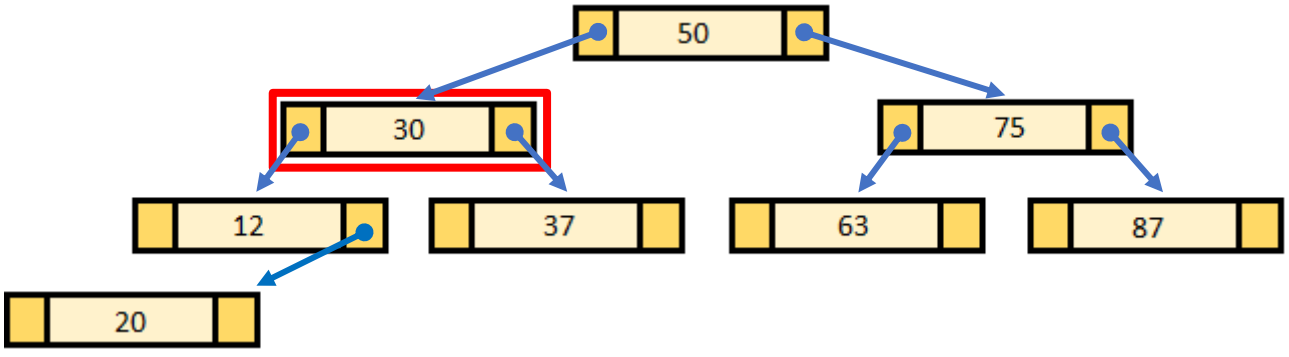
Marked the right node of the node that needed to delete (red rectangle, value = 37)

Then looks for the left node (if has)



Marked the left branch of the parent (red rectangle, value = 30)

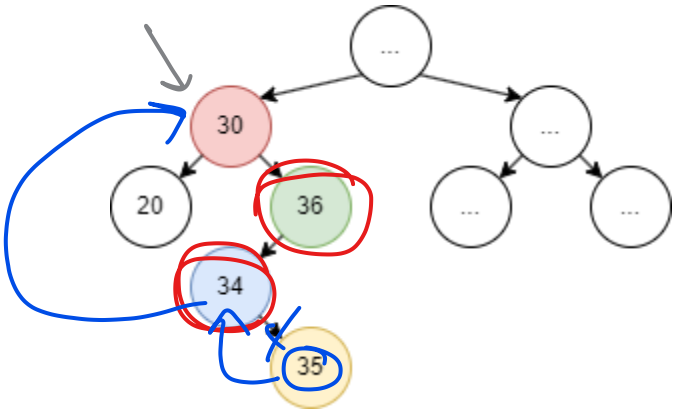
Delete the selected node and replace node with the new node



Delete node rules

- Always looks for the right of deleted node
- If the right node has sibling, check does it has left or right node.
- If has left node, use the left node do until no left node remains.
- If hasn't use the right node of deleted node

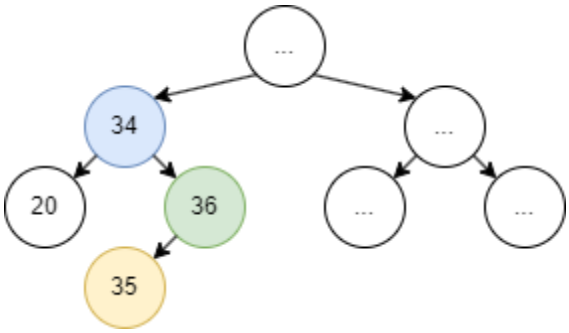
Delete node caution



Deleted node on binary tree

Declaration

- Red: The node that needed to delete
- Green: The children of the red node
- Blue: The children of the green node
- Yellow: The children of the blue node



Binary tree after deleted node

Exercise

Exercise 1: Forward and backward linked list

Create forward and backward linked list

- 1 <-> 2 <-> 3 <->4
- Draw data flow of forward and backward linked list

Exercise 2: Binary tree

Exercise 2.1: Create binary tree

- Insert node in order: 50,25,75,30,60,40,35,70,90,45,48,27,55,85,100
- Draw binary tree

Exercise 2.2: Remove value in binary tree

- Delete node: 30
- Delete node: 75
- Delete node: 35
- Draw binary tree for every node deleted

Exercise 2.3: Theory part (coding is needed)

- Find maximum height
- Find parent (node that contain next left or right node)
- Find children (node that contain previous node)
- Find leaves (node that does not contain next left or right node)
- Find sibling (node that has the same parent)