Data Structures
Tree – Binary Search Tree

Team Emertxe

Binary Search Tree -Delete an element

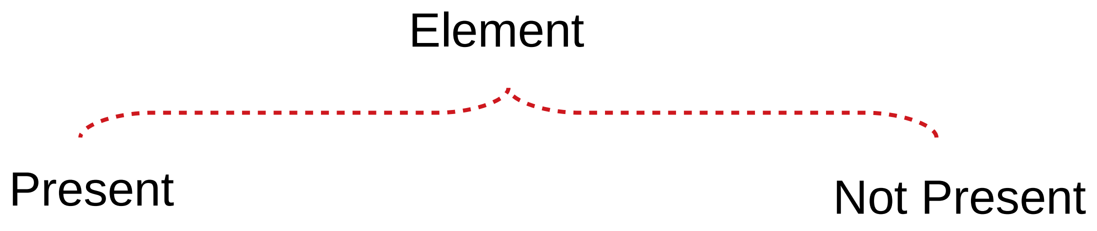
Data Structure – Binary Search Tree

Delete Element

Cases :

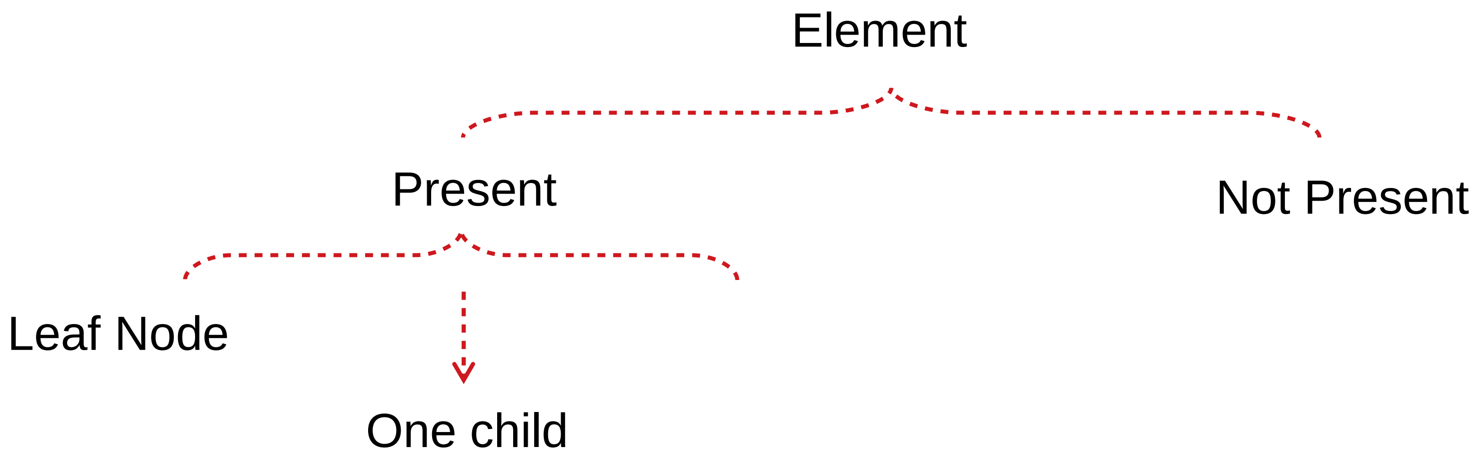
Delete Element

Cases :



Delete Element

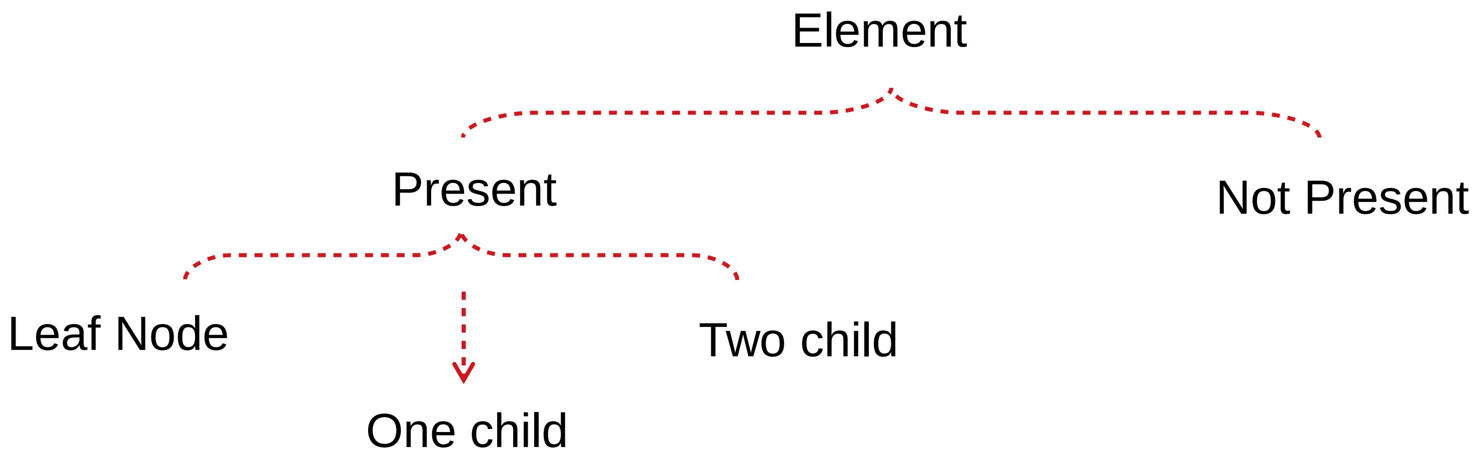
Cases :



Data Structure – Binary Search Tree

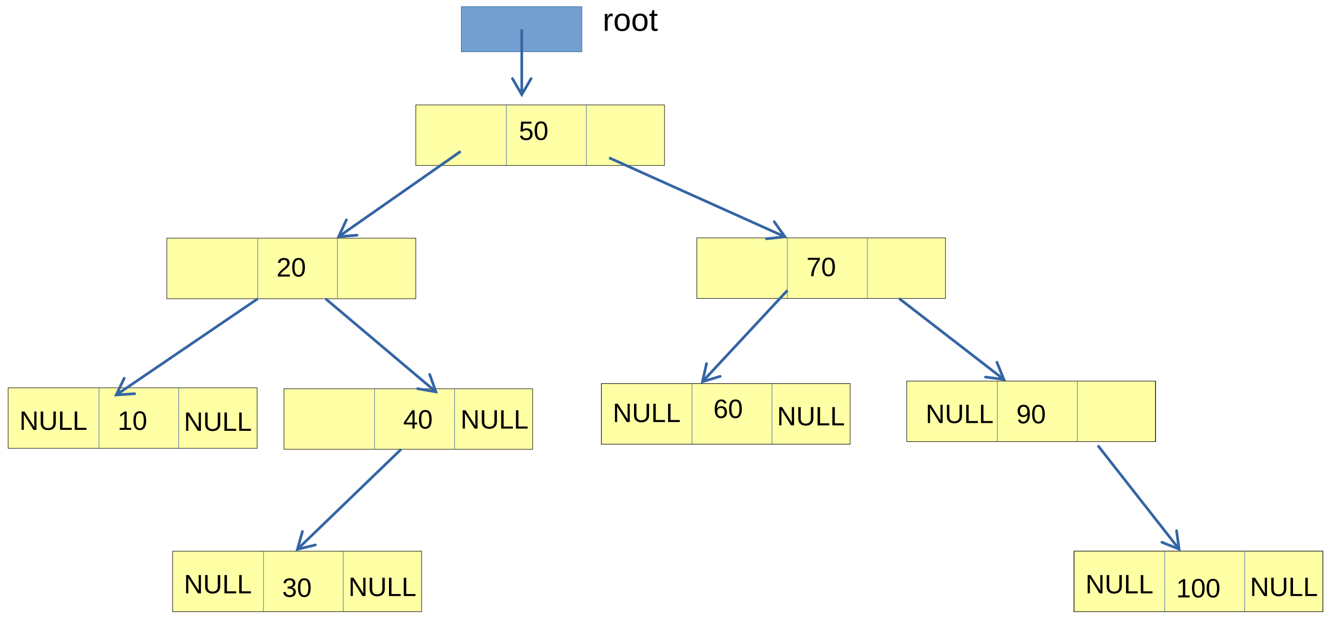
Delete Element

Cases :



Data Structure – Binary Search Tree

Delete Element

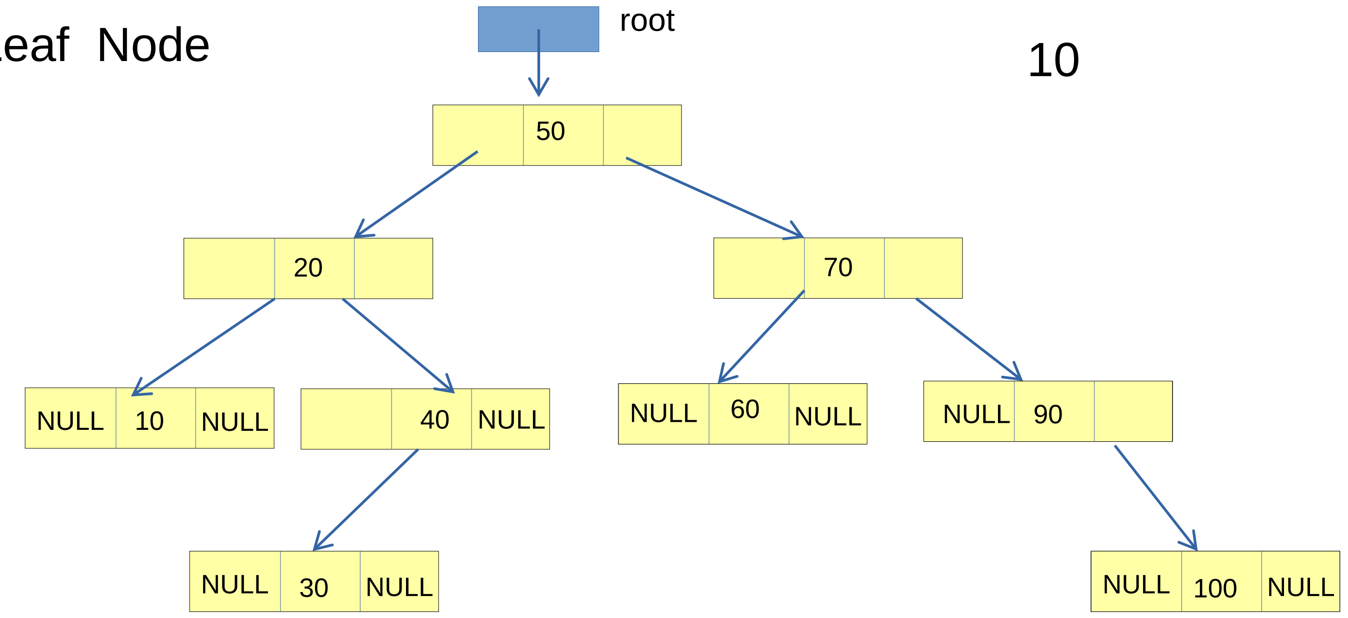


Data Structure – Binary Search Tree

Delete Element

Case : Leaf Node

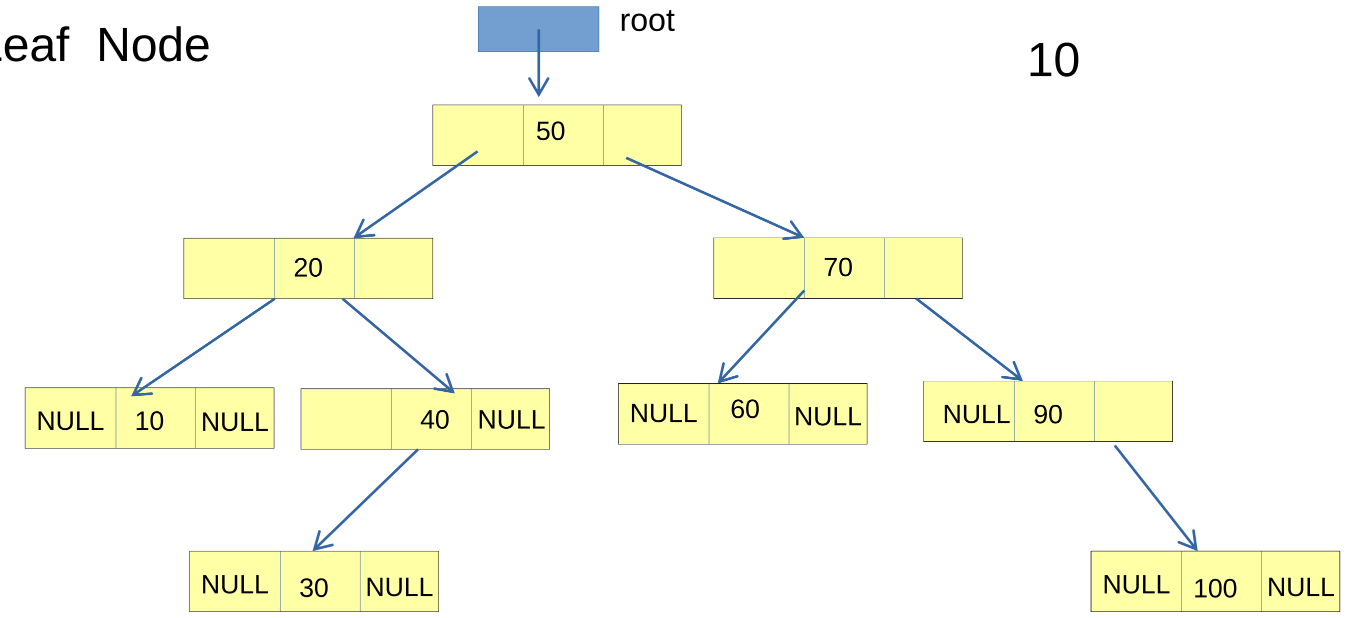
10



Data Structure – Binary Search Tree

Delete Element

Case : Leaf Node



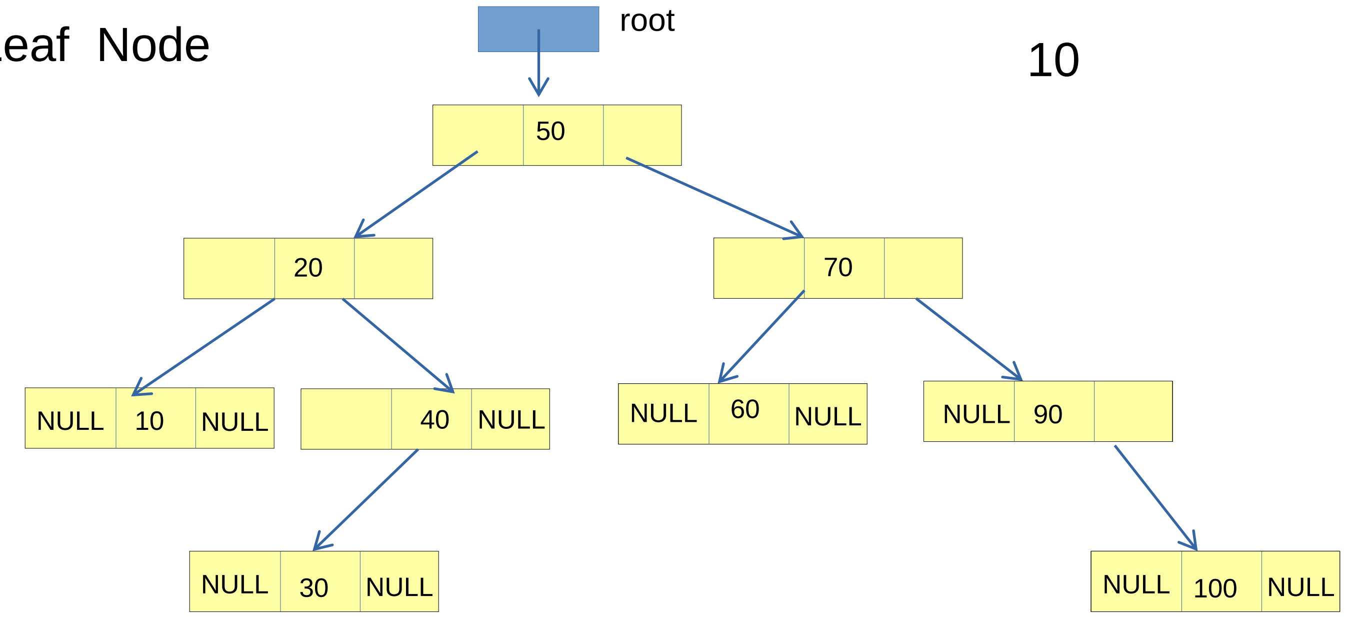
10

Data Structure – Binary Search Tree

Delete Element

Case : Leaf Node

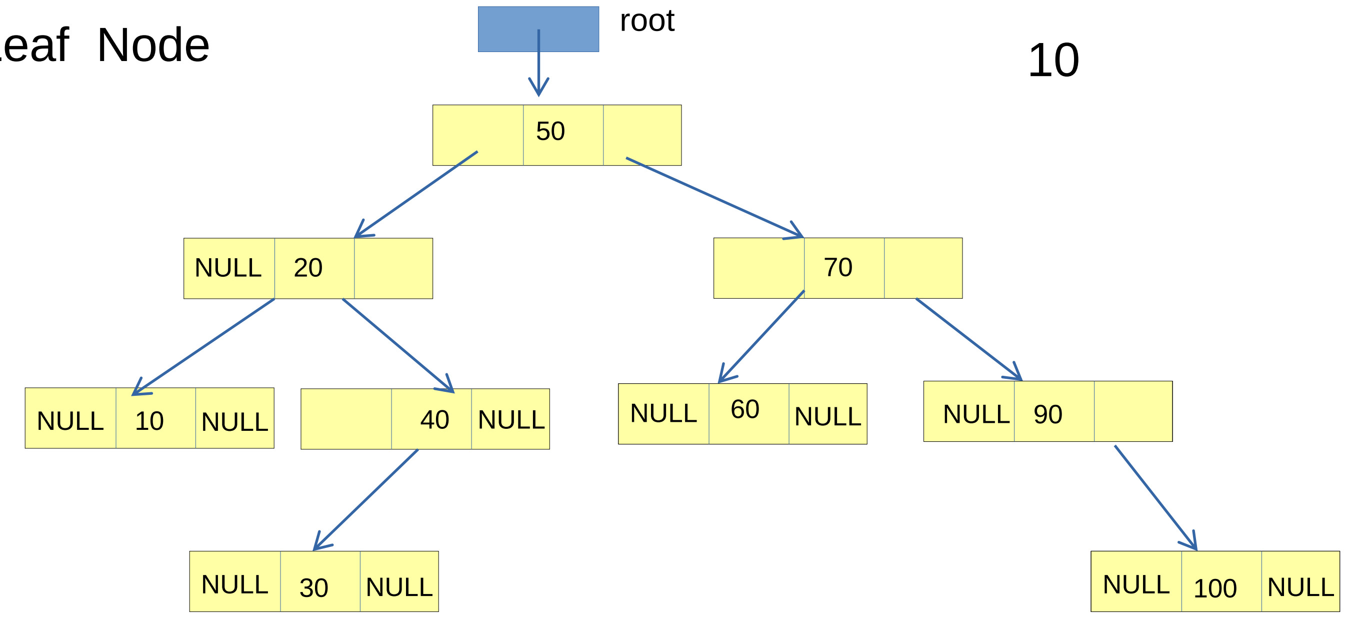
10



Data Structure – Binary Search Tree

Delete Element

Case : Leaf Node



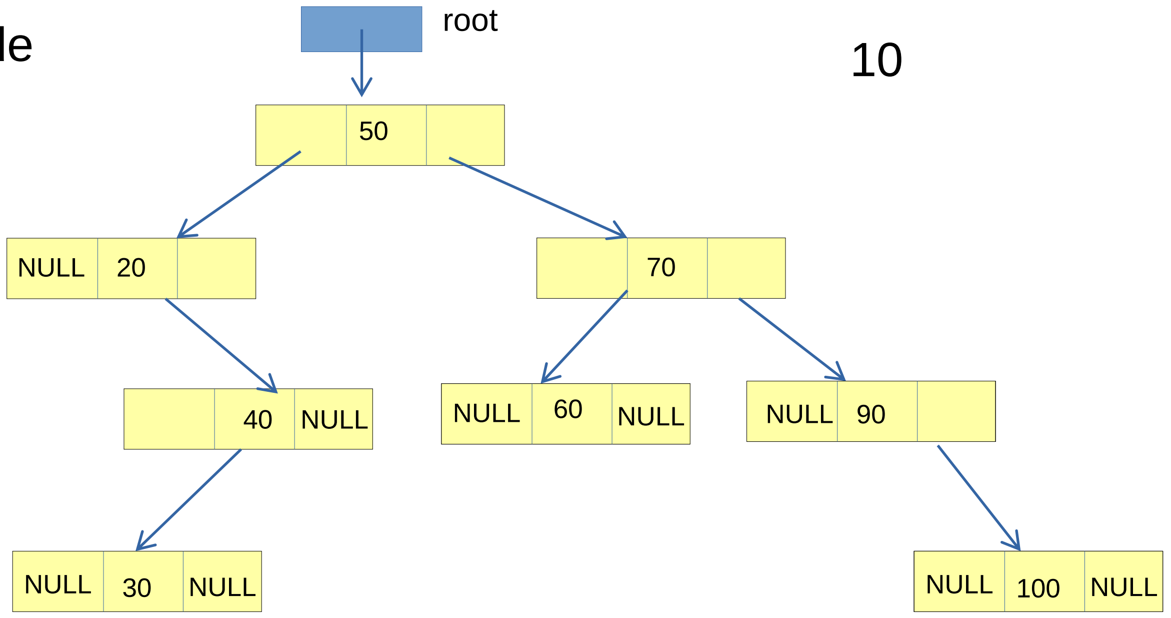
10

Data Structure – Binary Search Tree

Delete Element

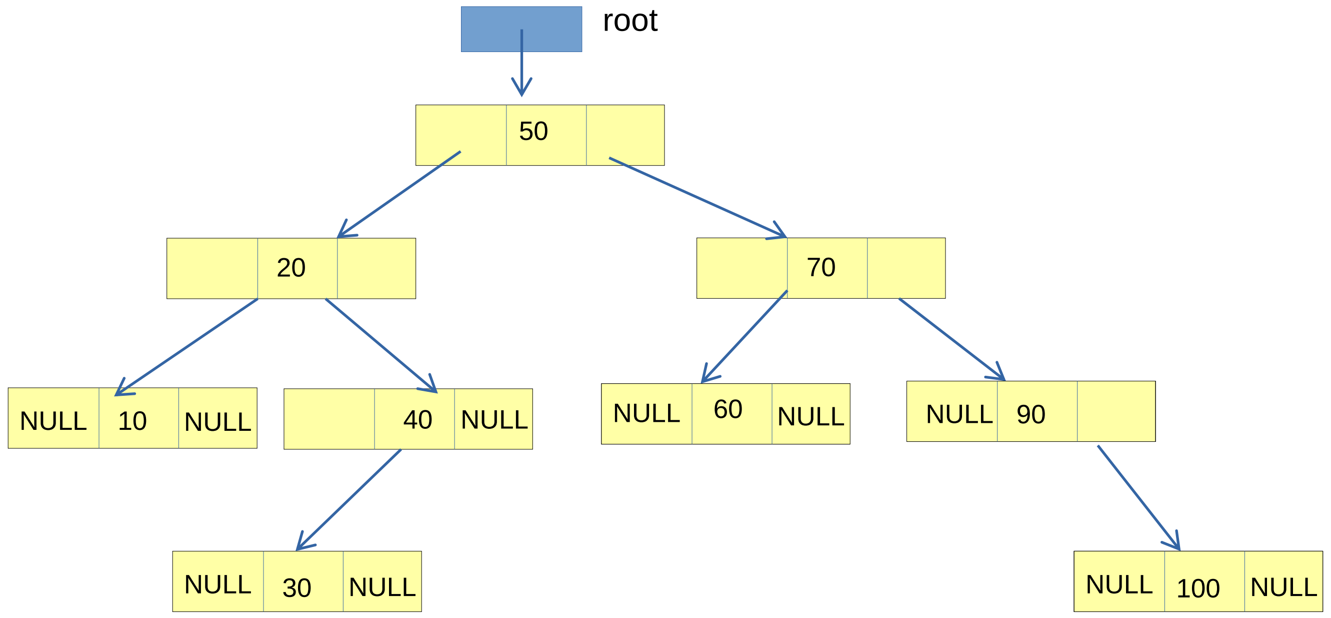
Case : Leaf Node

10



Data Structure – Binary Search Tree

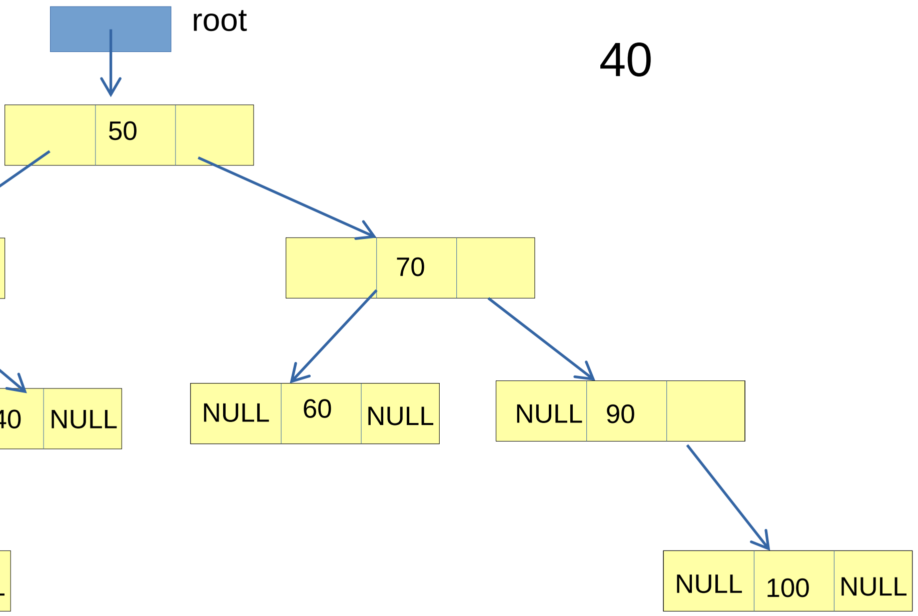
Delete Element



Data Structure – Binary Search Tree

Delete Element

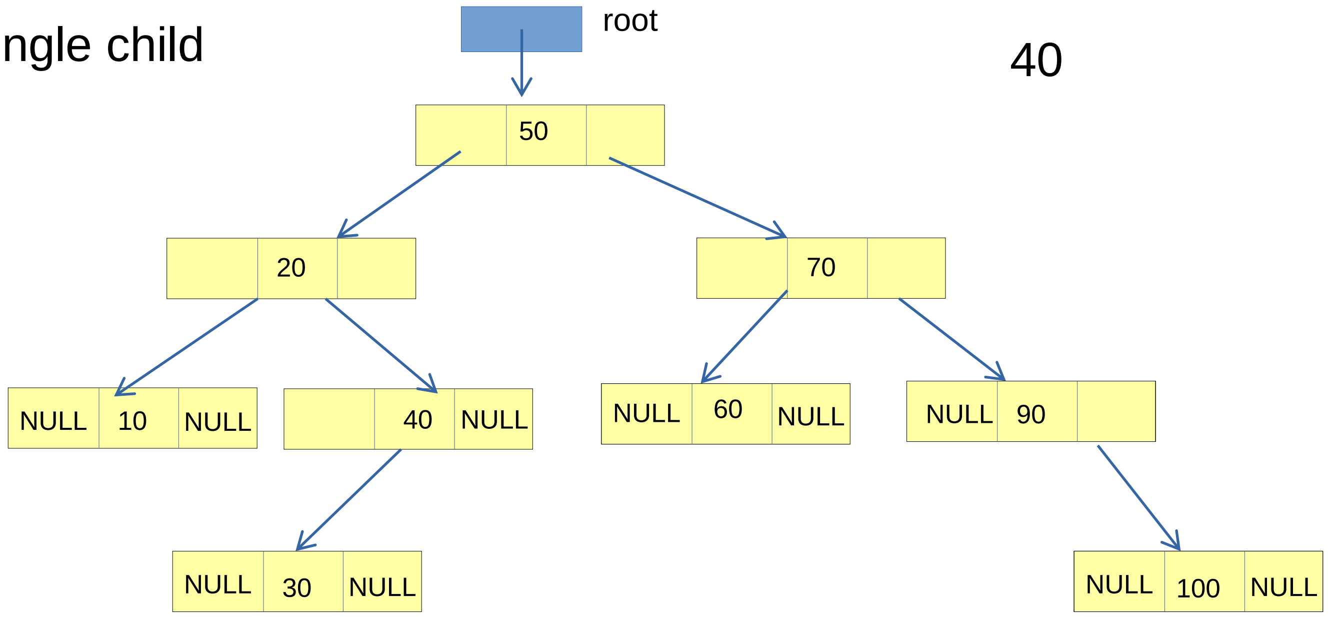
Case : Single child



40

Delete Element

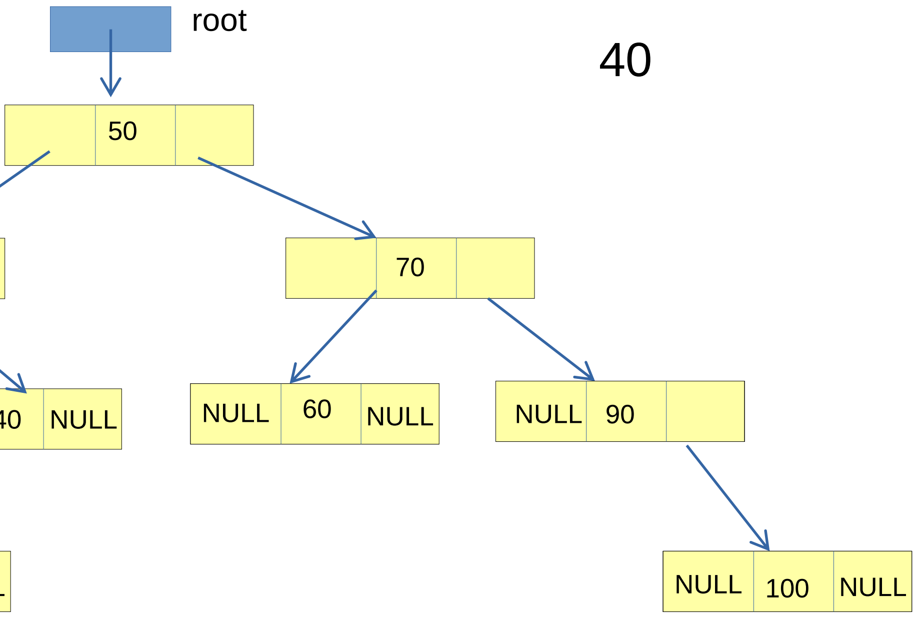
Case : Single child



40

Delete Element

Case : Single child



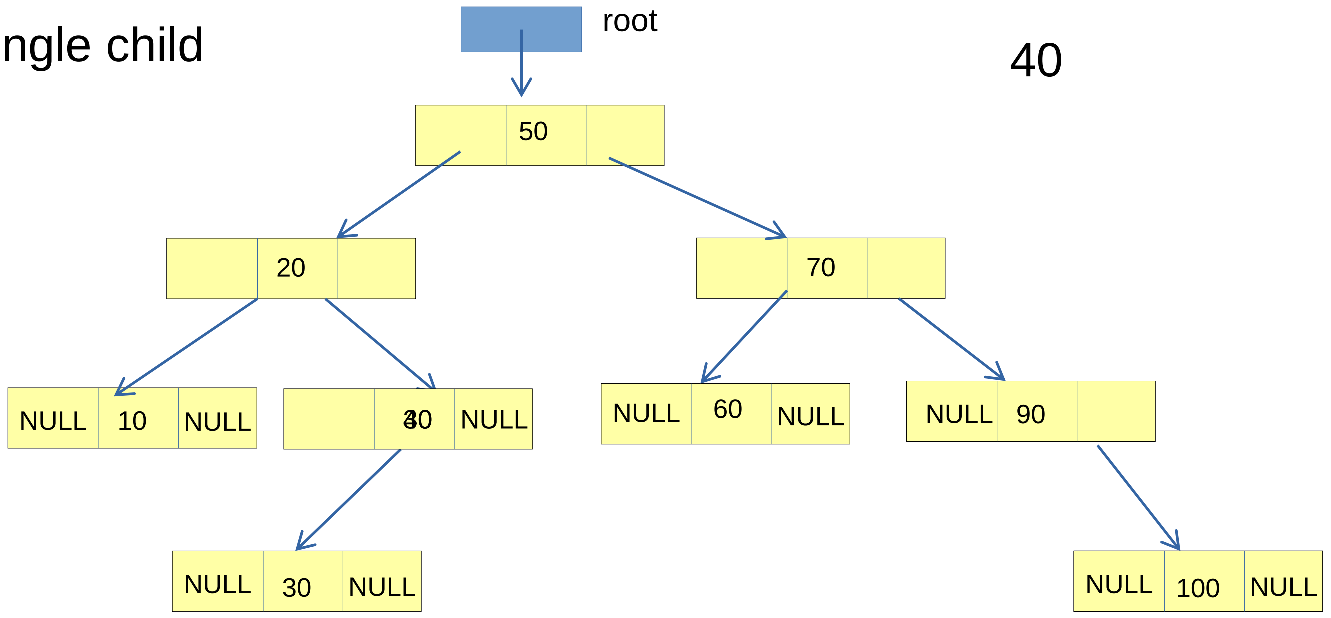
40

Data Structure – Binary Search Tree

Delete Element

Case : Single child

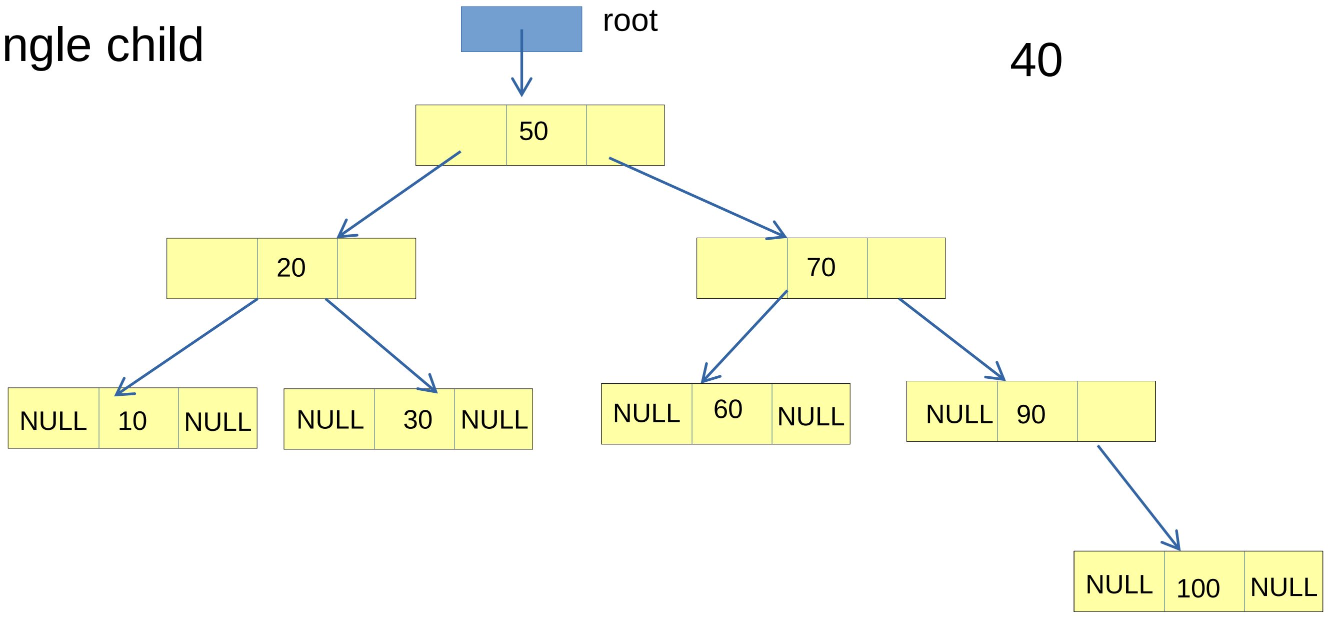
40



Delete Element

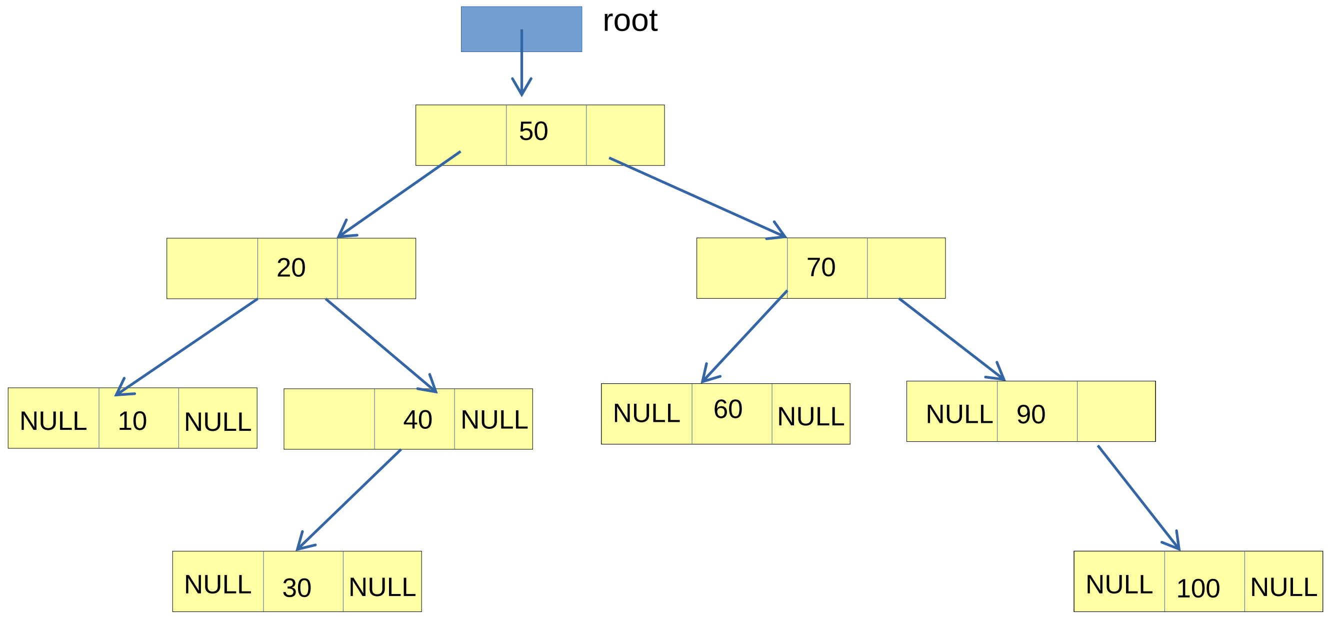
Case : Single child

40



Data Structure – Binary Search Tree

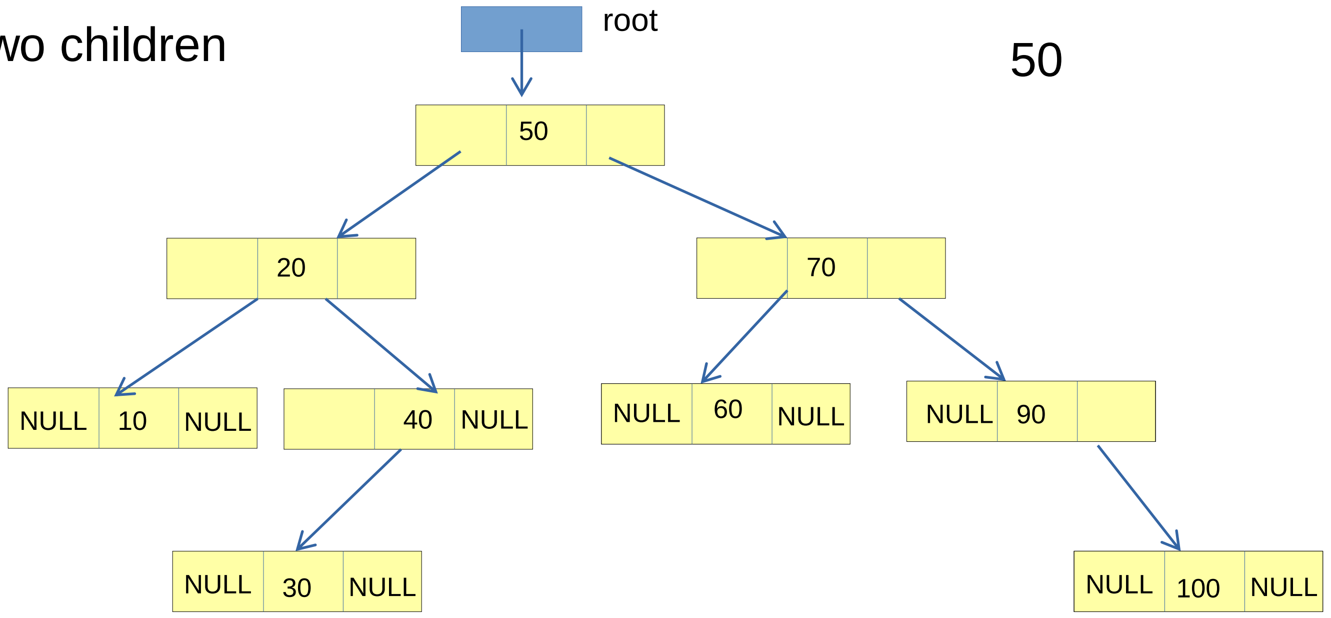
Delete Element



Data Structure – Binary Search Tree

Delete Element

Case : Two children

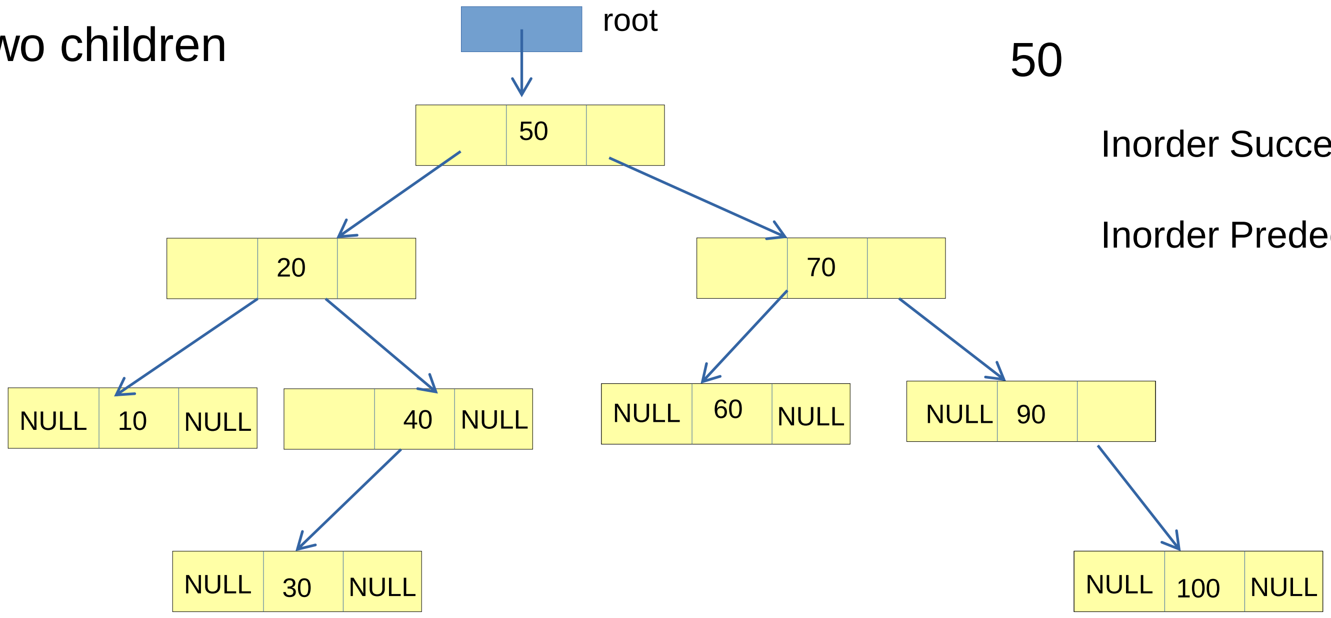


50

Data Structure – Binary Search Tree

Delete Element

Case : Two children



50

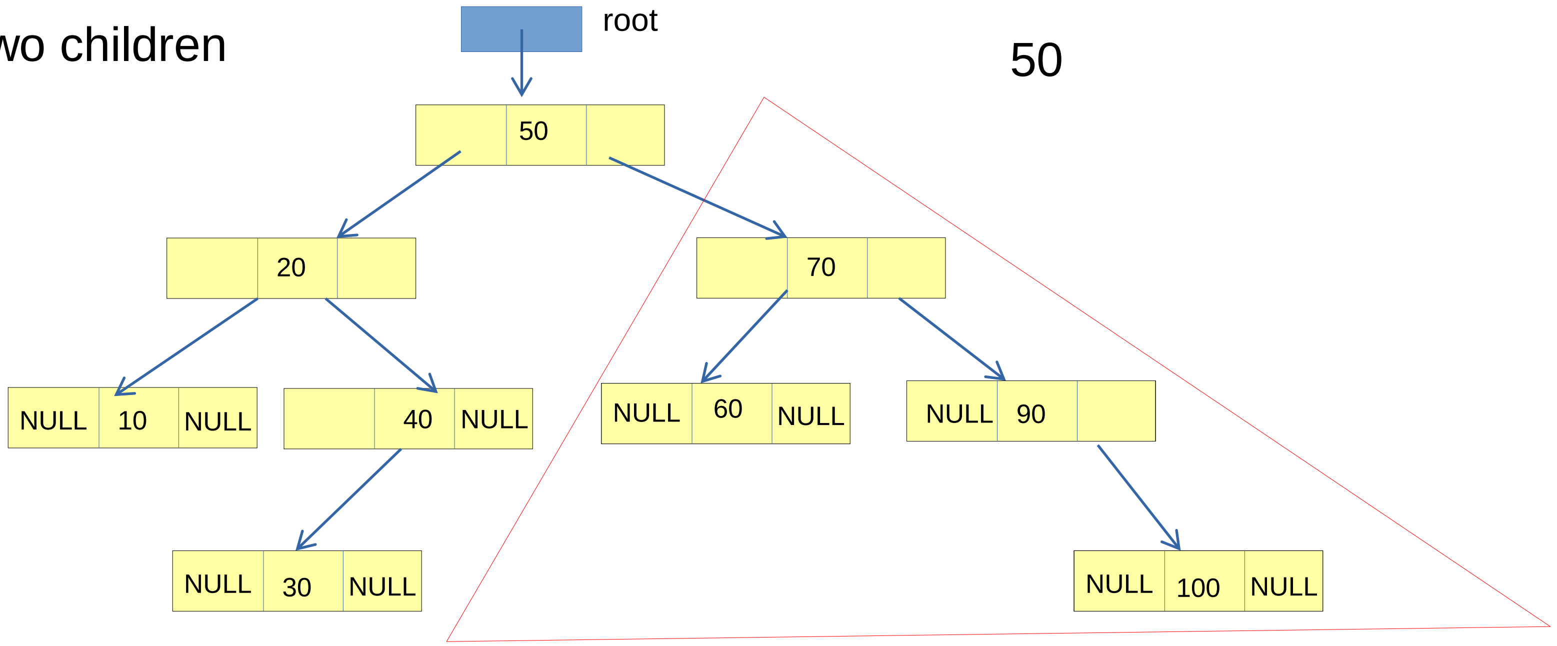
Inorder Successor

Inorder Predecessor

Data Structure – Binary Search Tree

Delete Element

Case : Two children

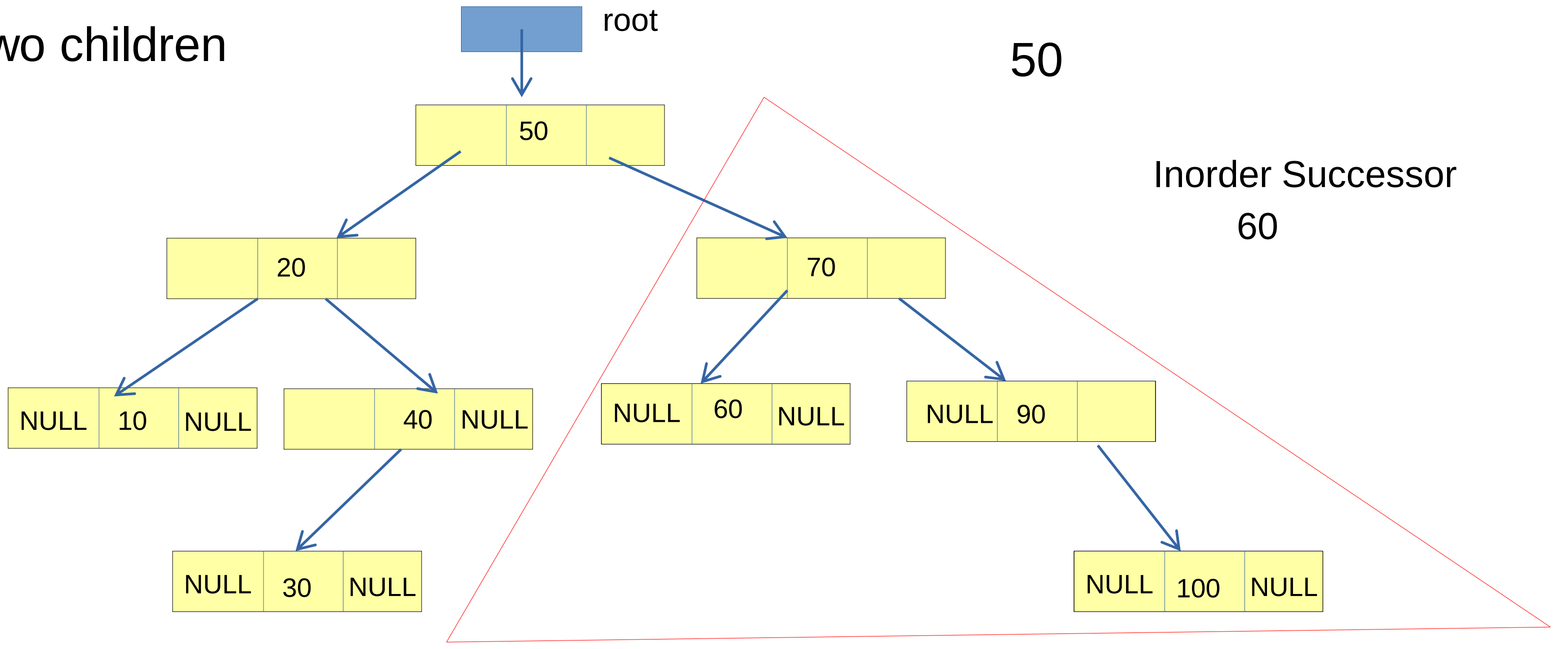


50

Data Structure – Binary Search Tree

Delete Element

Case : Two children



50

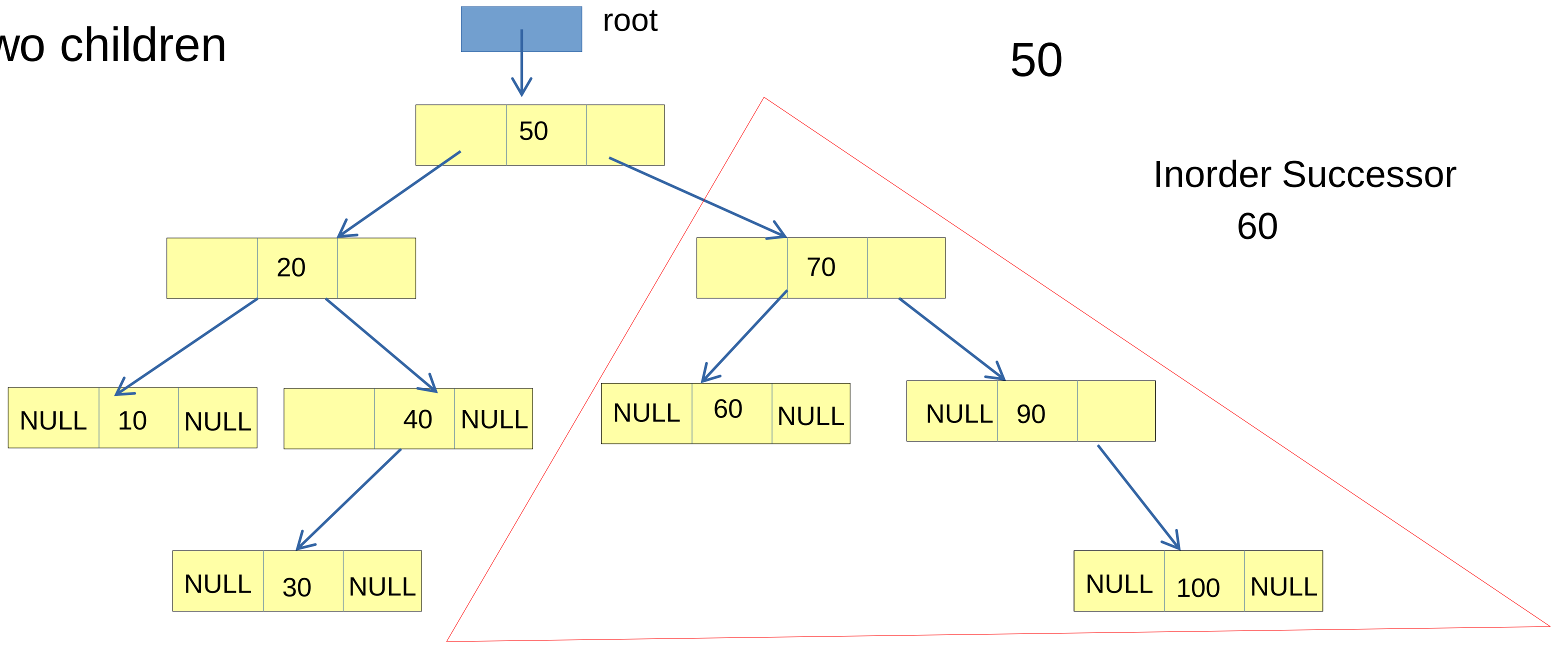
Inorder Successor

60

Data Structure – Binary Search Tree

Delete Element

Case : Two children



50

Inorder Successor

60

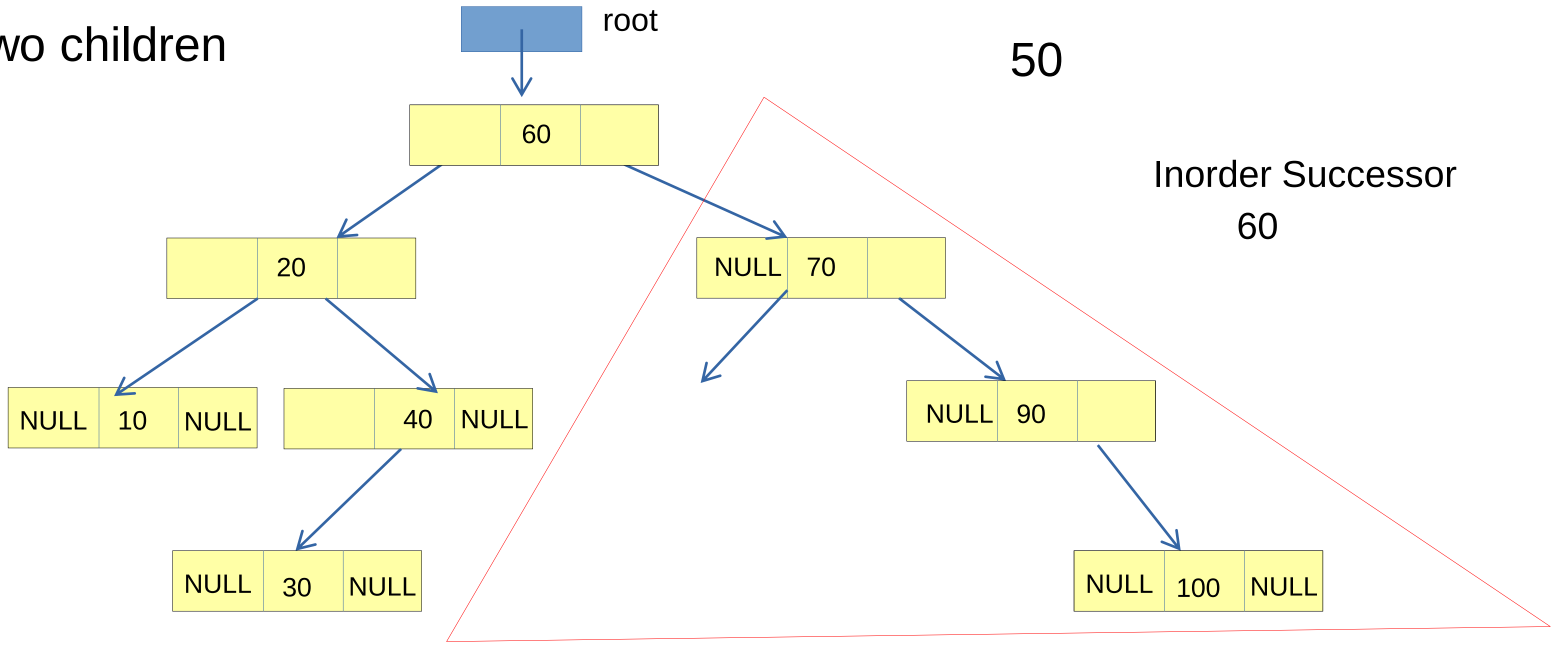
Delete Element

Case : Two children

50

Inorder Successor

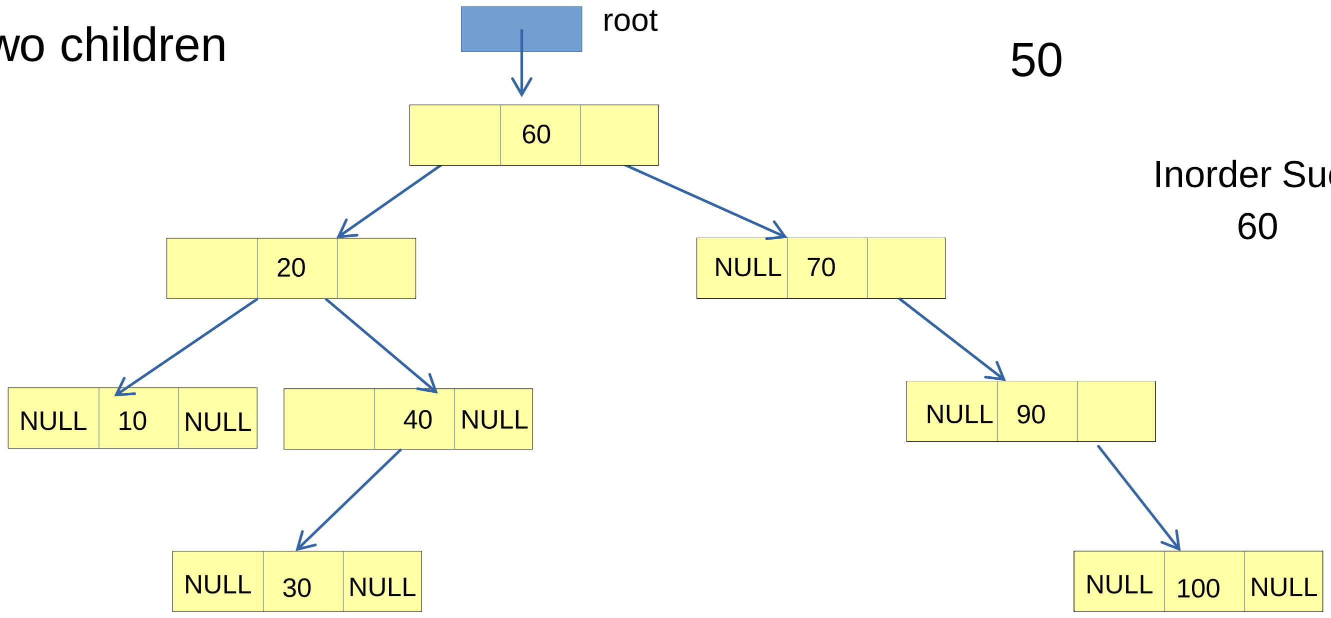
60



Data Structure – Binary Search Tree

Delete Element

Case : Two children



50

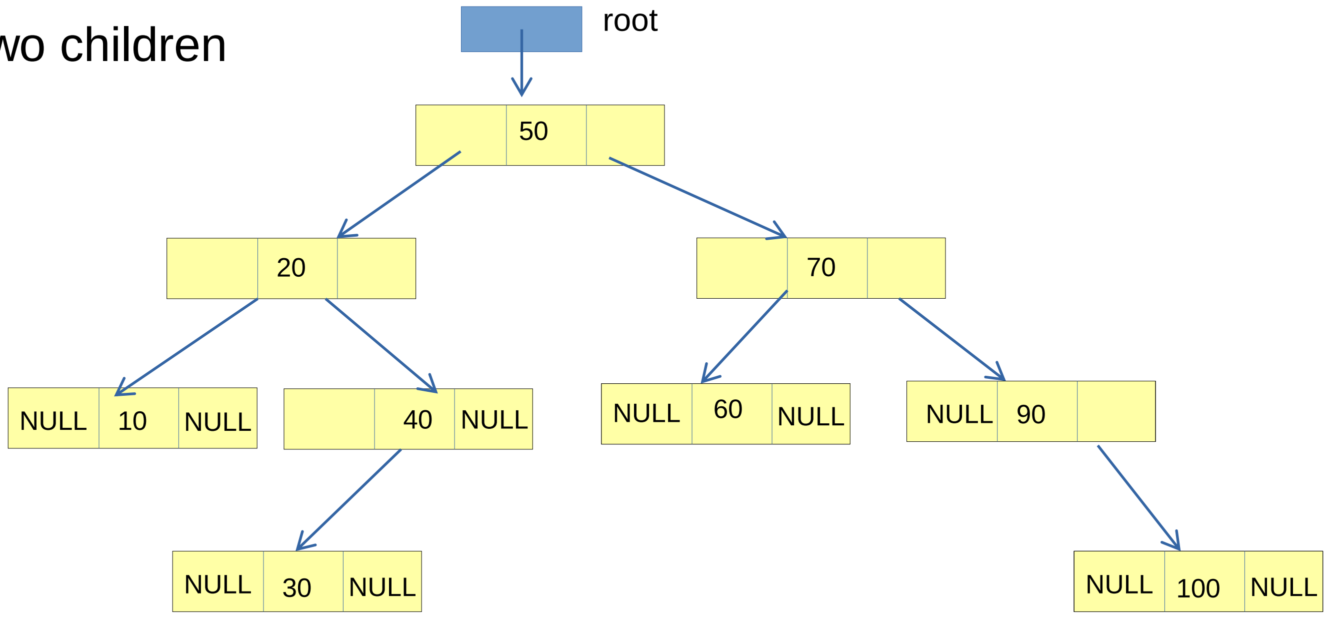
Inorder Successor

60

Data Structure – Binary Search Tree

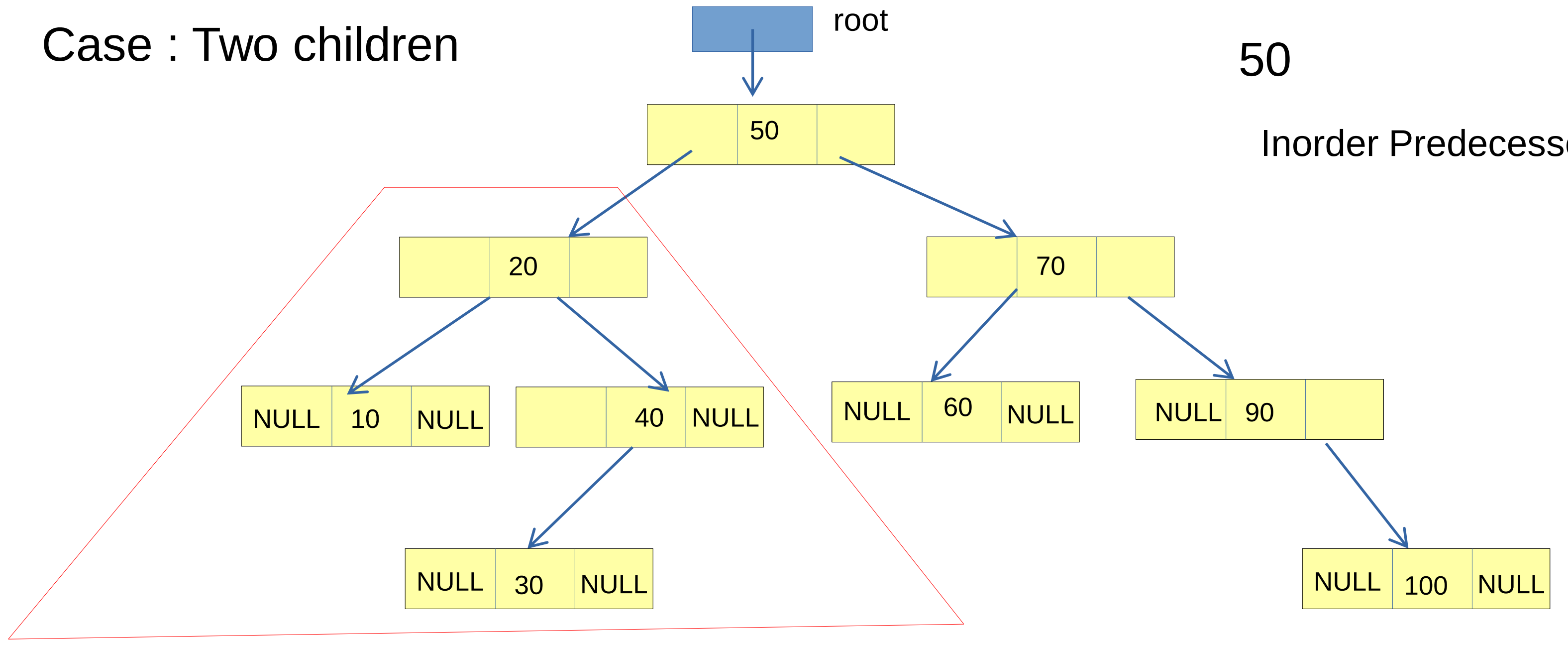
Delete Element

Case : Two children



Delete Element

Case : Two children

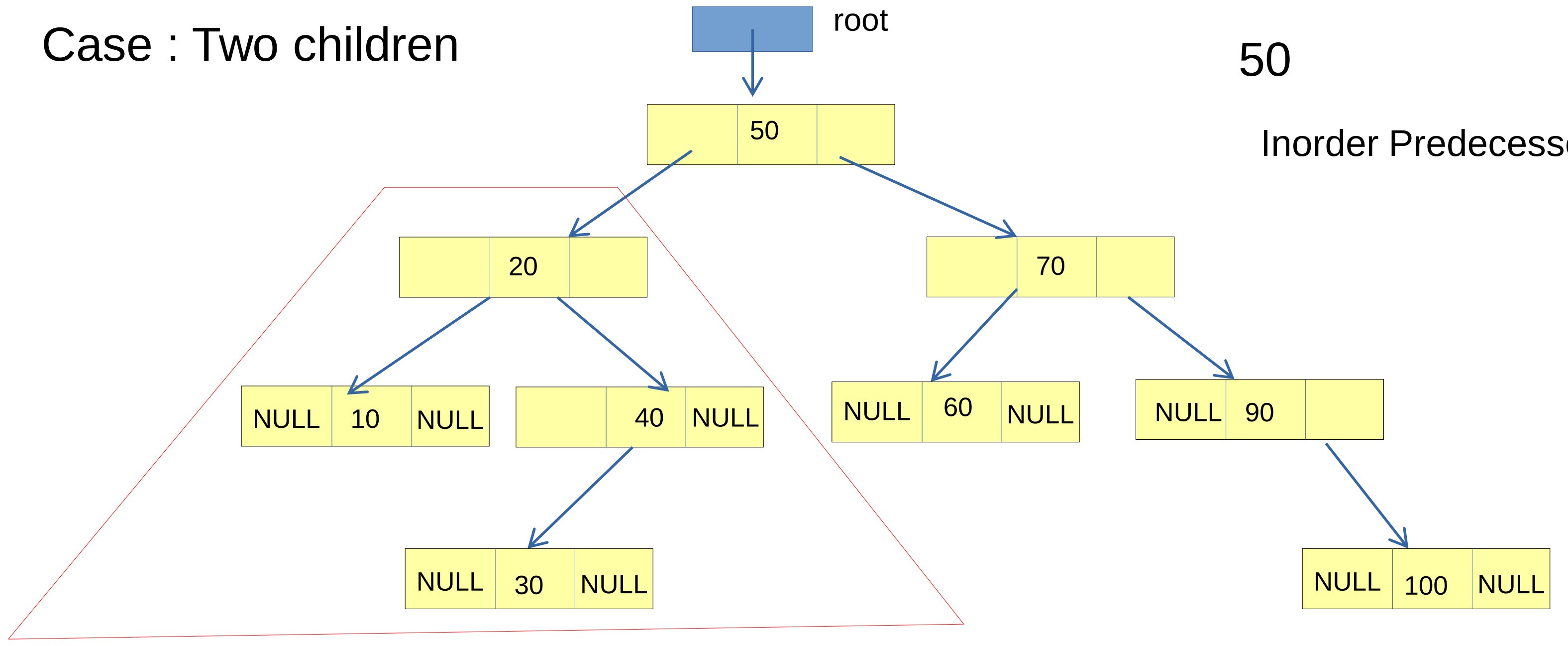


50

Inorder Predecessor

Delete Element

Case : Two children



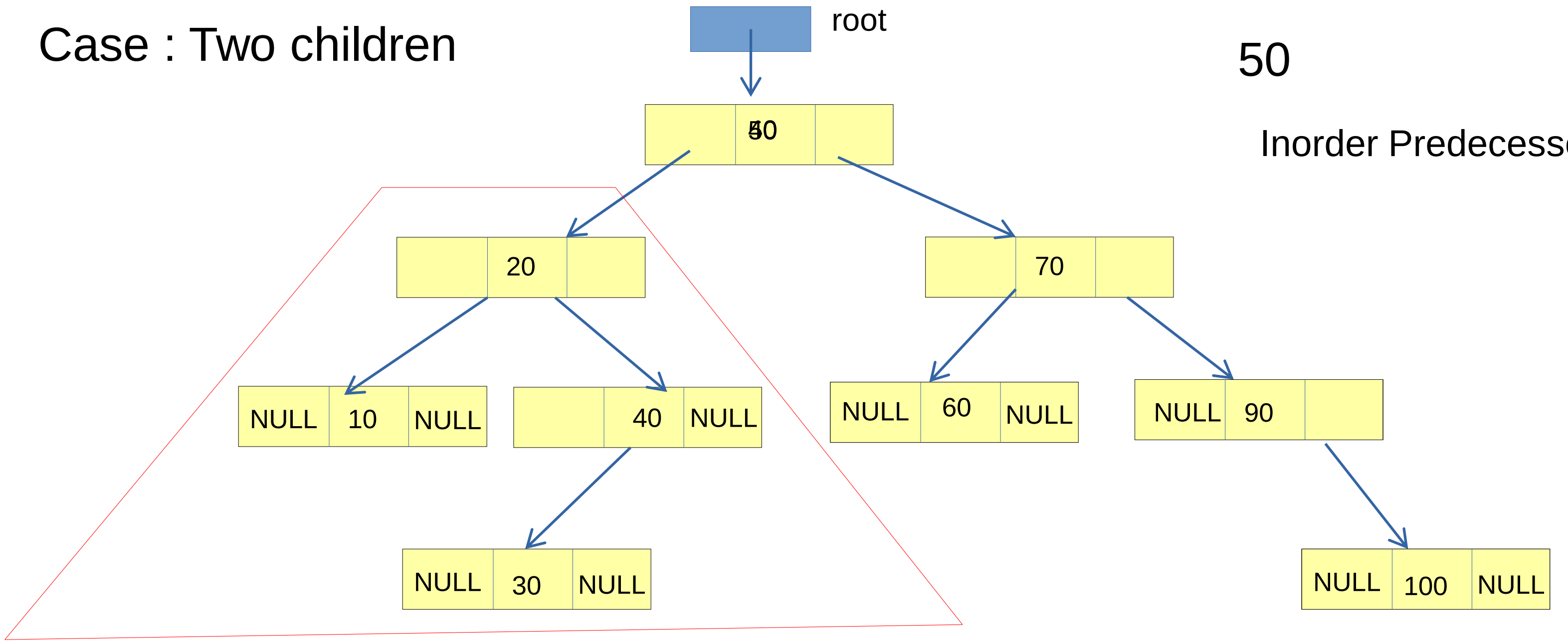
50

Inorder Predecessor 40

Data Structure – Binary Search Tree

Delete Element

Case : Two children



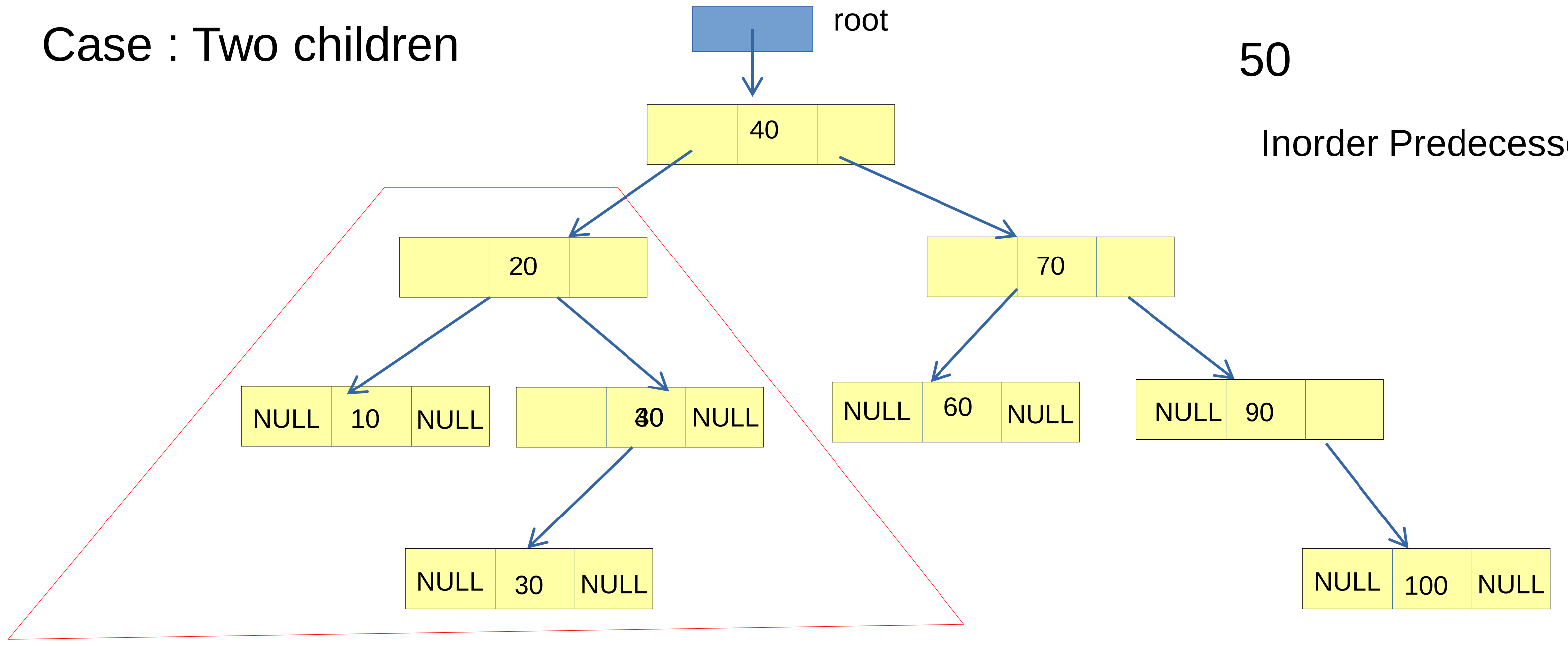
50

Inorder Predecessor 40

Data Structure – Binary Search Tree

Delete Element

Case : Two children

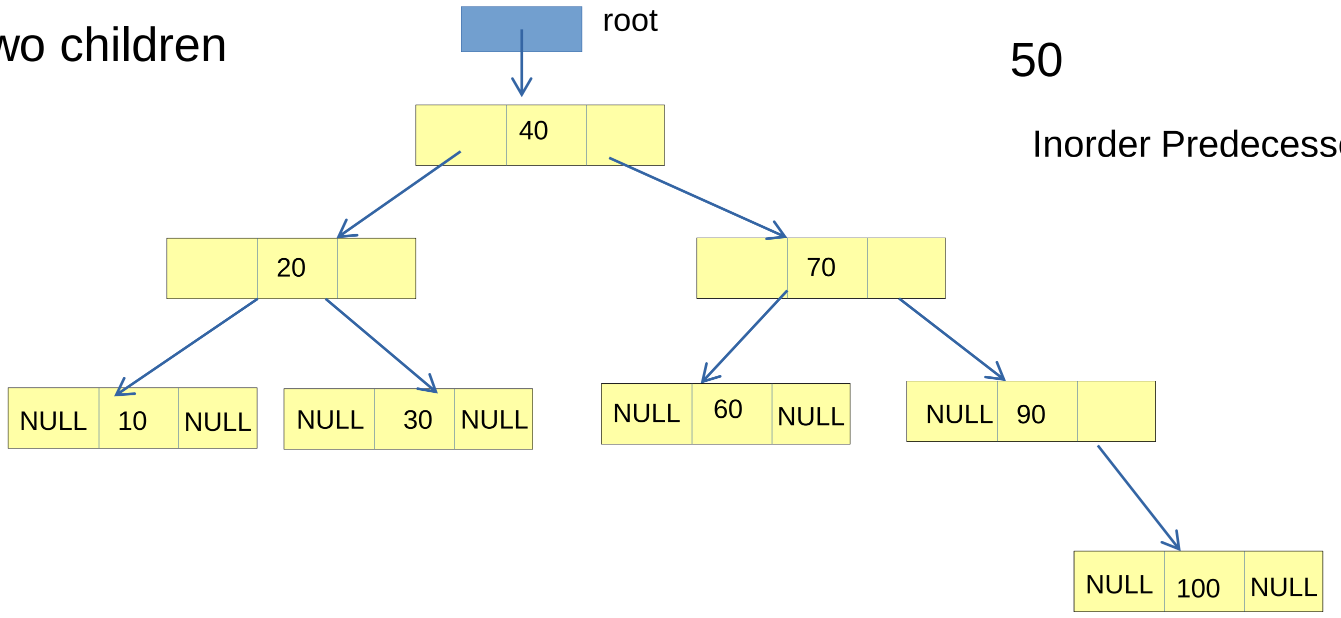


50

Inorder Predecessor 40

Delete Element

Case : Two children



50

Inorder Predecessor 40

Algorithm : delete_node(root,key)

```
If (root = NULL)
        return root
If (key < root ->data )
        root -> LC = delete_node(root -> LC ,key)
Else if (key > root -> data )
        root -> RC = delete_node(root -> RC ,key)
Else
        If (root -> LC = NULL)
                temp = root -> RC
                free(root)
                return temp
        Else if (root -> RC = NULL)
                temp = root -> LC
                free(root)
                return temp
        Else
                temp = min_node(root -> RC)
                root -> data = temp -> data
                root -> RC = delete_node(root -> RC,temp->data)
return root
```

Algorithm : delete_node(root,key)

```
If (root = NULL)
        return root
If (key < root ->data )
        root -> LC = delete_node(root -> LC ,key)
Else if (key > root -> data )
        root -> RC = delete_node(root -> RC ,key)
Else
        If (root -> LC = NULL)
                temp = root -> RC
                free(root)
                return temp
        Else if (root -> RC = NULL)
                temp = root -> LC
                free(root)
                return temp
        Else
                temp = min_node(root -> RC)
                root -> data = temp -> data
                root -> RC = delete_node(root -> RC,temp->data)
return root
```

min_node(root)

```
temp = root
while (temp  AND temp -> LC !=NULL)
        temp = temp ->LC
return temp
```

Code - delete_node(root,key)