Trees (3)

Insert Node

and Delete Node

الدكتور
اثير العاني

Insert Node

Given a sorted list:

5 8 9 13 21 44 45 46

Insert Node

Given a sorted list:

5 8 9 13 21 44 45 46

If <u>14</u> is inserted, the list become:

5 8 9 13 <u>14</u> 21 44 45 46

Insert Node

Given a sorted list:

5 8 9 13 21 44 45 46

If <u>14</u> is inserted, the list become:

5 8 9 13 14 21 44 45 46

Insert Node

Given a sorted list:

5 8 9 13 21 44 45 46

If <u>14</u> is inserted, the list become:

5 8 9 13 14 21 44 45 46

<u>FEATURES OF A SORTED LIST</u> IS PRESERVED

Insert Node

Given a sorted list:

5 8 9 13 21 44 45 46

If <u>14</u> is inserted, the list become:

5 8 9 13 14 21 44 45 46

<u>FEATURES OF A SORTED LIST</u> IS PRESERVED

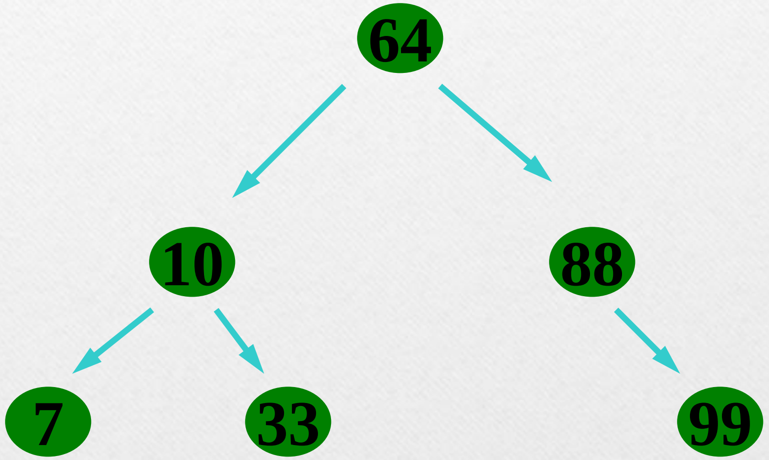
WHAT ARE THE FEATURES OF A SORTED LIST????

Insert Node

Given a BST:

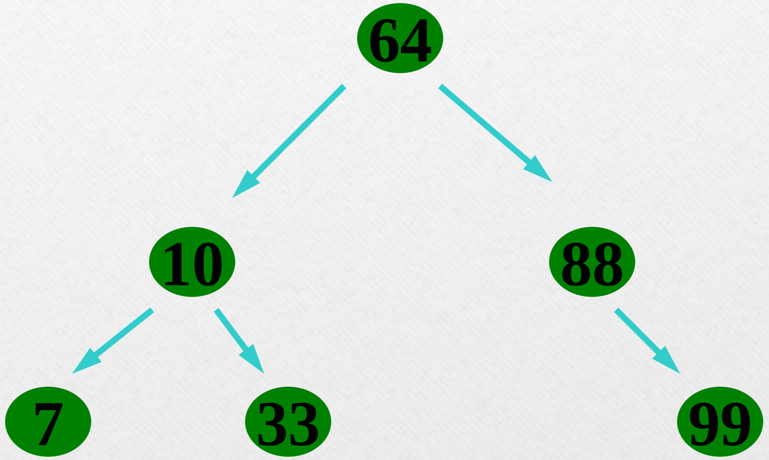
Insert Node

Given a BST:



Insert Node

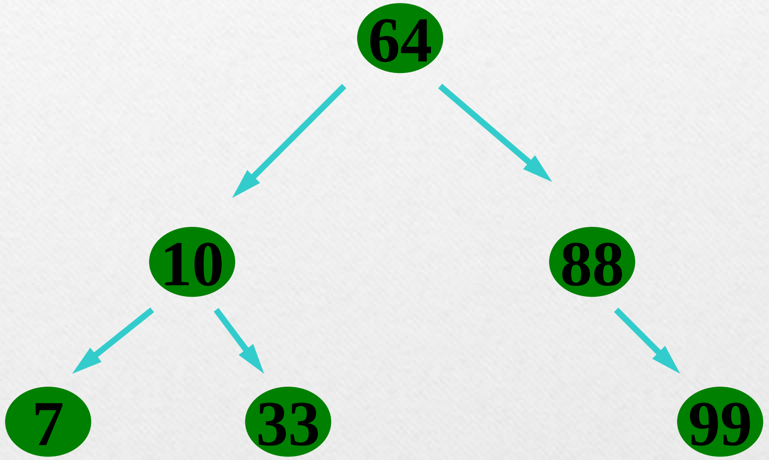
Given a BST:



If 14 to be inserted :

Insert Node

Given a BST:

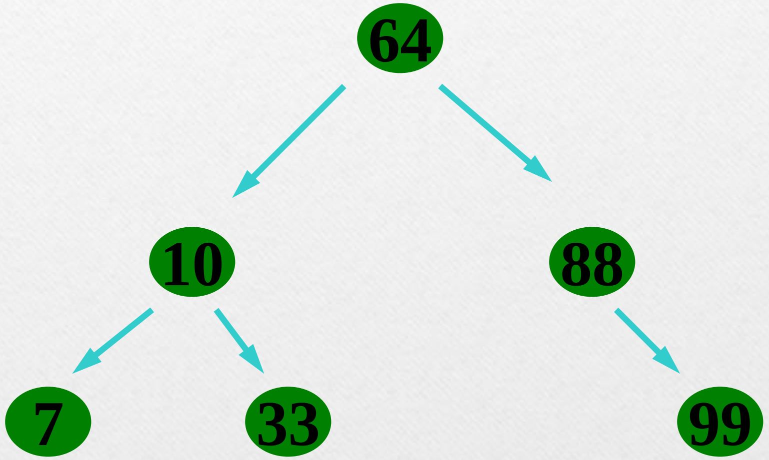


If 14 to be inserted :

How does the new BST look like ?

Insert Node

Given a BST:

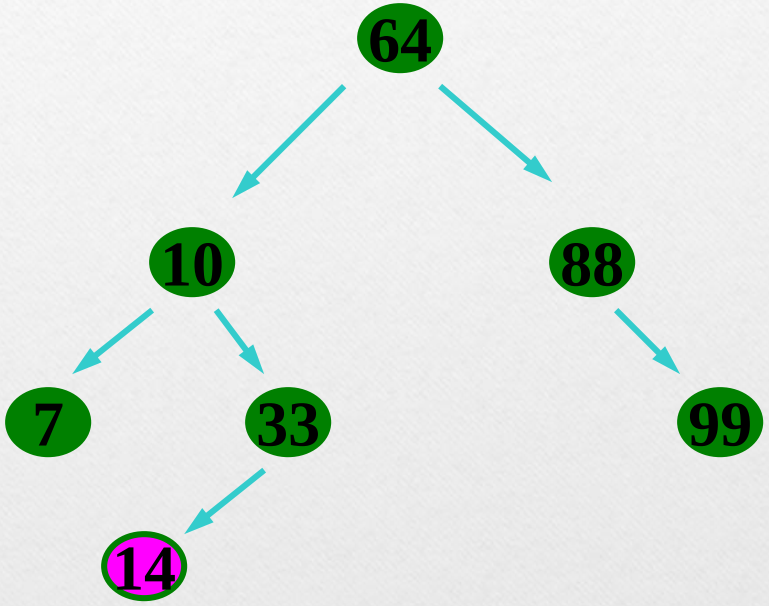


If 14 to be inserted :

How does the new BST look like ?
How do you do it?

Insert Node

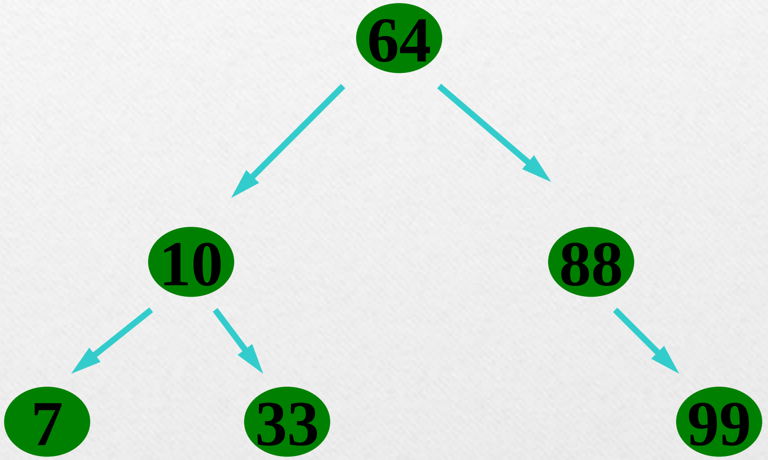
Given a BST:



Are the features of a BST preserved???

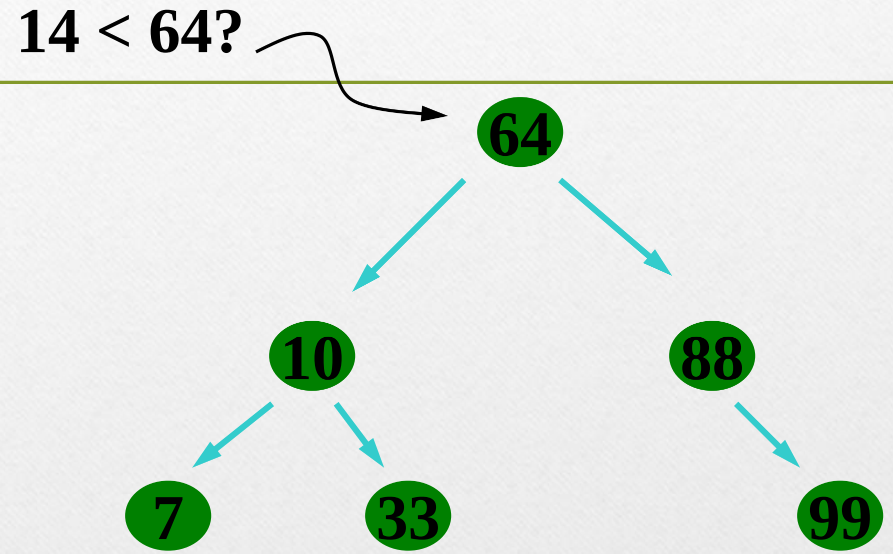
Algorithm to insert 14:

Insert Node



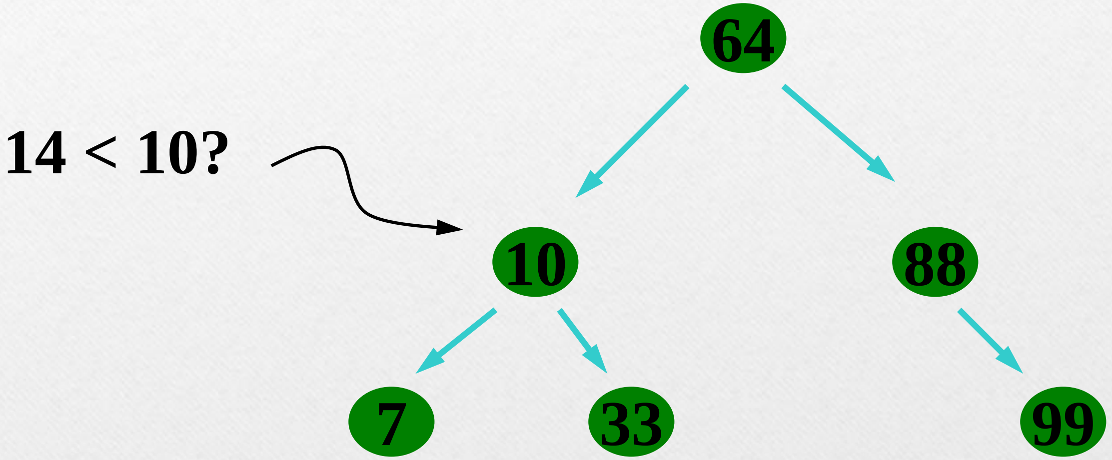
Algorithm to insert 14:

Insert Node



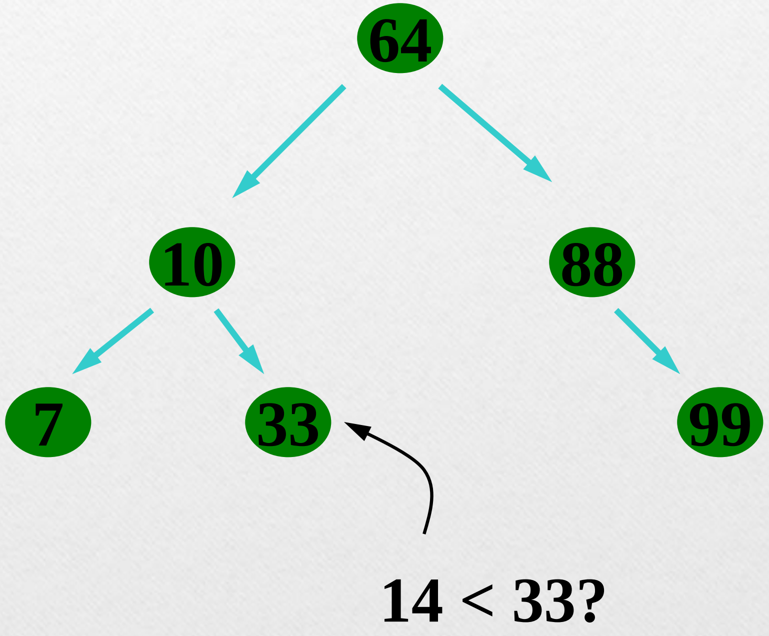
Algorithm to insert 14:

Insert Node



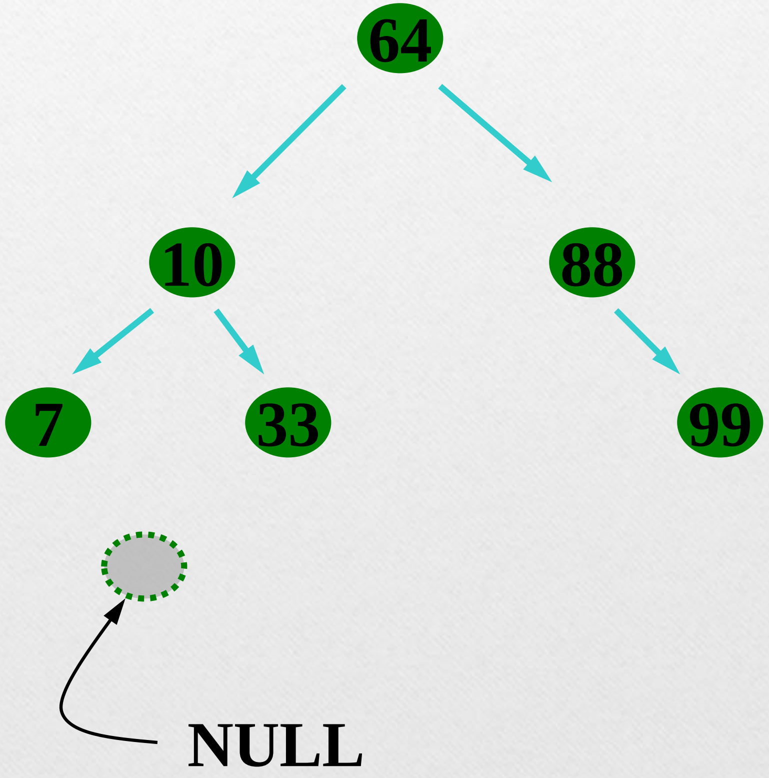
Algorithm to insert 14:

Insert Node



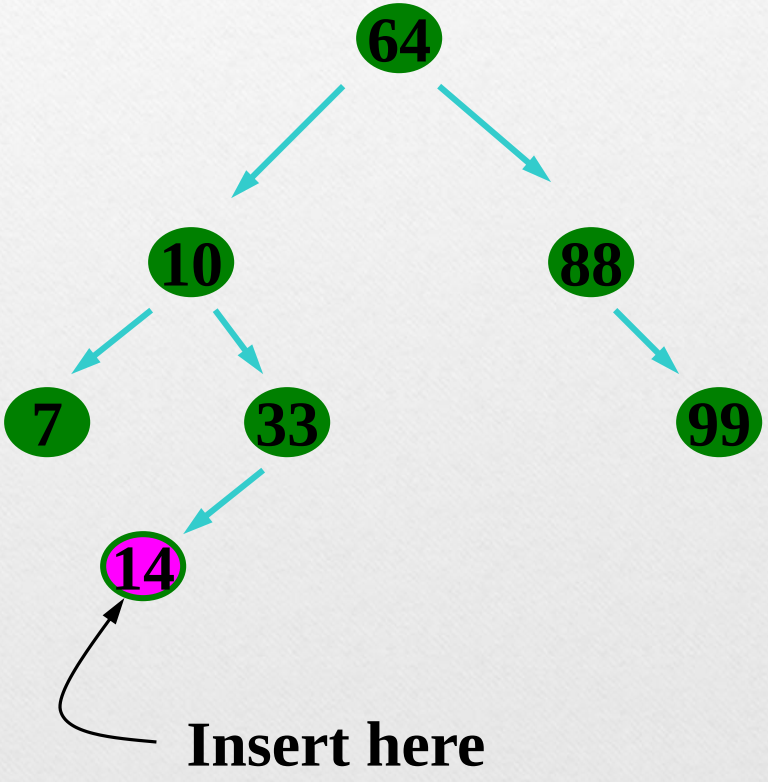
Algorithm to insert 14:

Insert Node



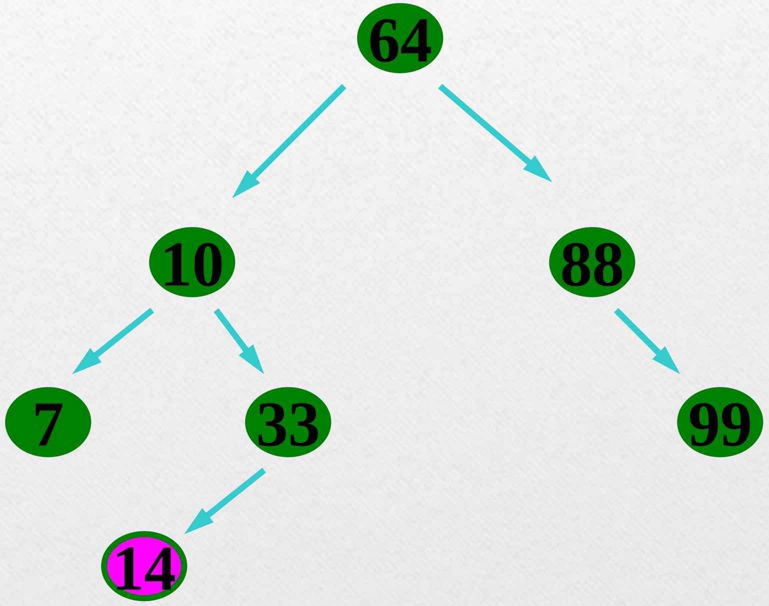
Algorithm to insert 14:

Insert Node



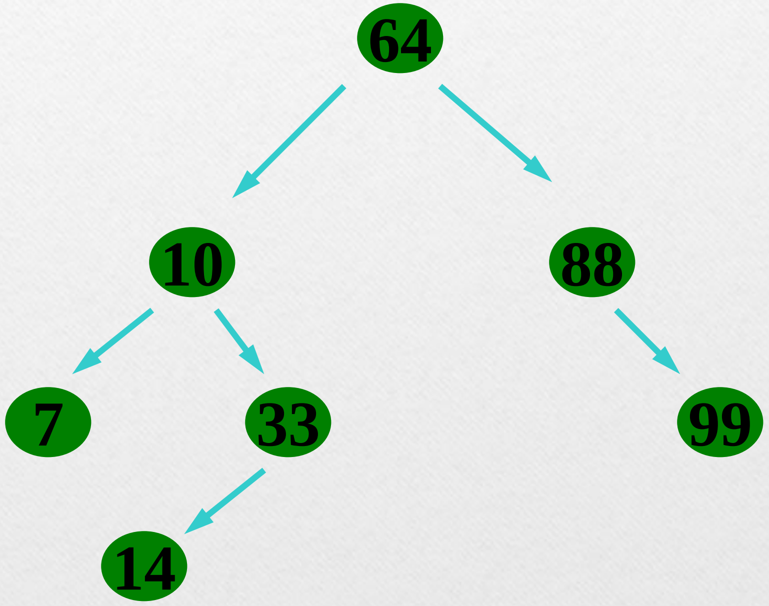
Algorithm to insert 14:

Insert Node



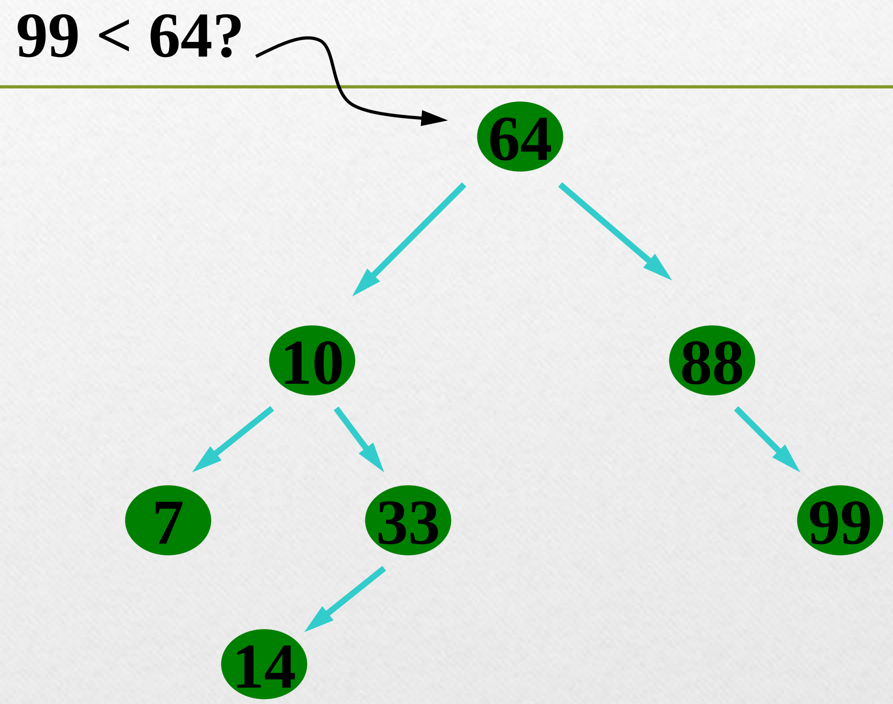
Try insert 99:

Insert Node



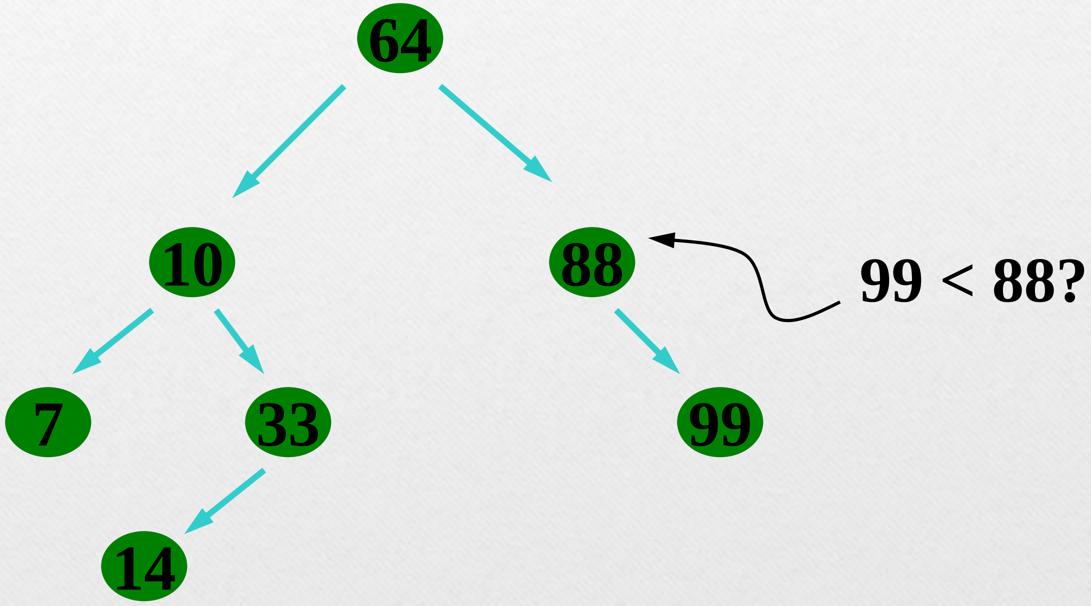
Try insert 99:

Insert Node



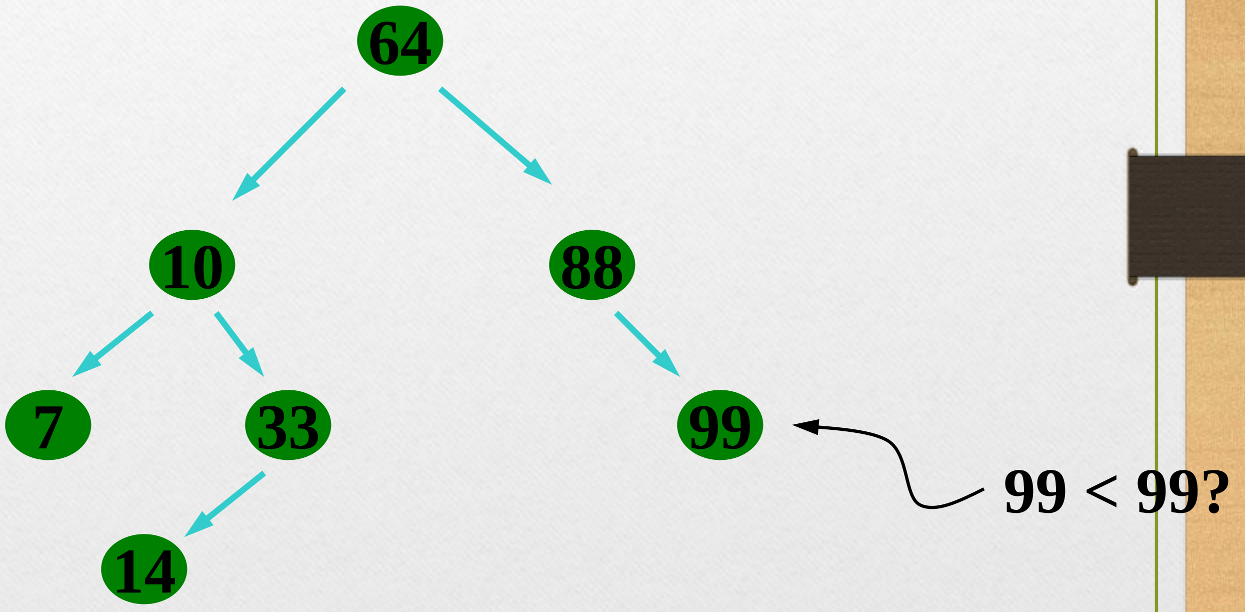
Try insert 99:

Insert Node



Try insert 99:

Insert Node



Insert Node

This insert algorithm is not very effective

Try building a BST using the Same algorithm for the following Sequence:

K N G I C U

Insert Node

This insert algorithm is
not very effective

Try building a BST using the
Same algorithm for the following
Sequence:

K N G I C U

K U C I N G

Insert Node

This insert algorithm is not very effective

Try building a BST using the Same algorithm for the following Sequence:

K N G I C U
K U C I N G
C G I K N U

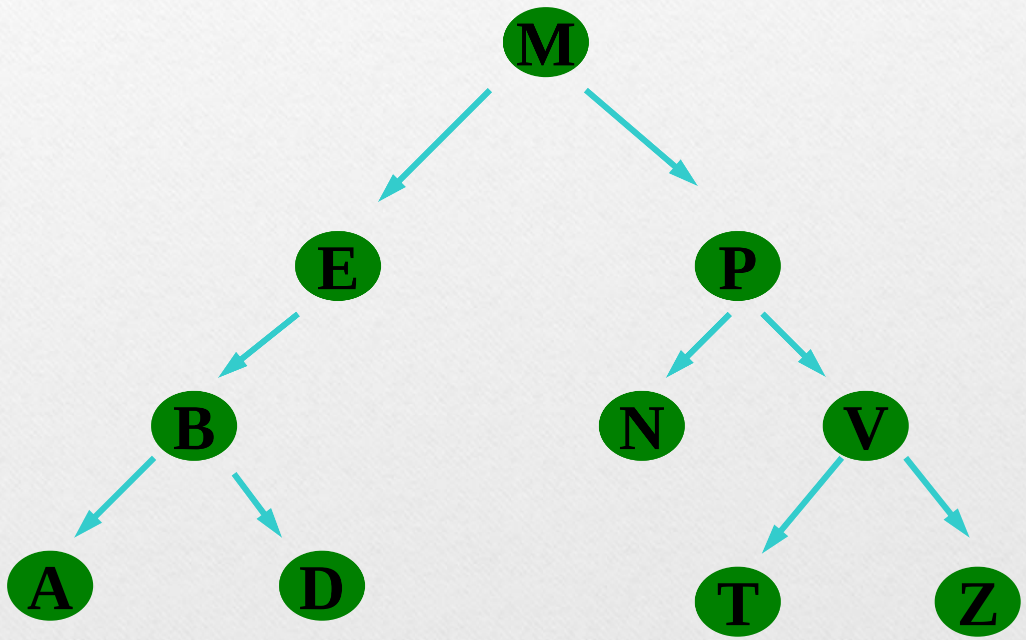
Delete Node

3 cases node deletion :

- The node to be deleted is a leaf
- The node to be deleted has 1 child
- The node to be deleted has 2 children

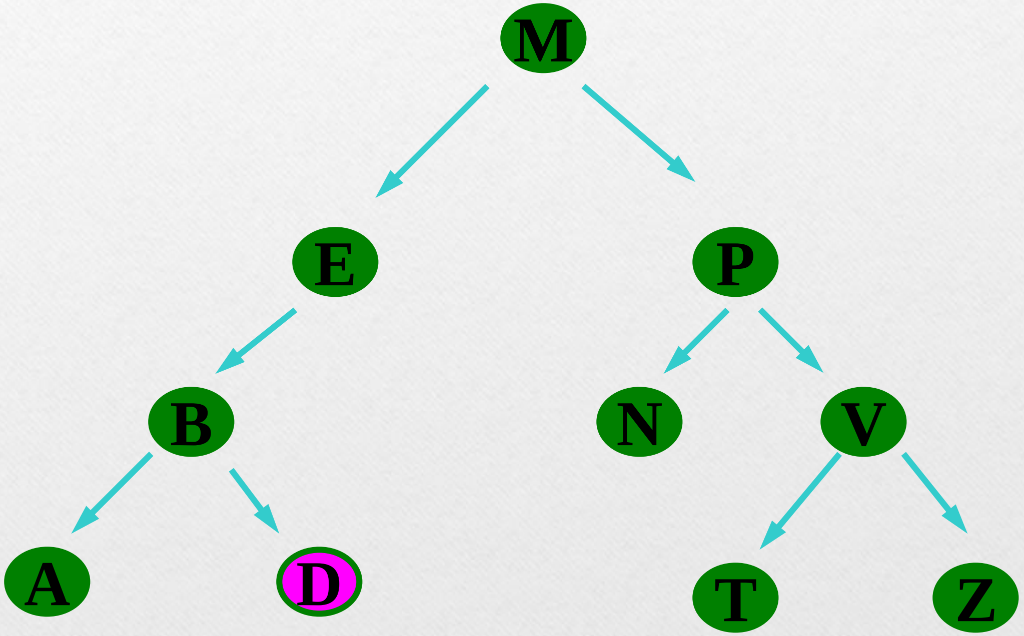
Delete Node: First Case

To delete D:



Delete Node: First Case

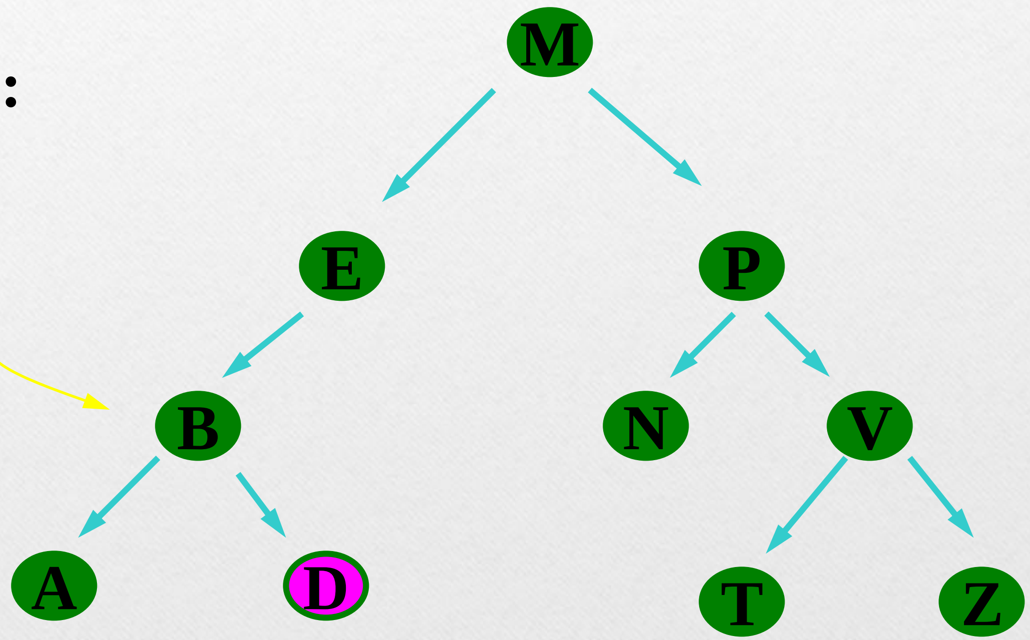
To delete D:



Delete Node: First Case

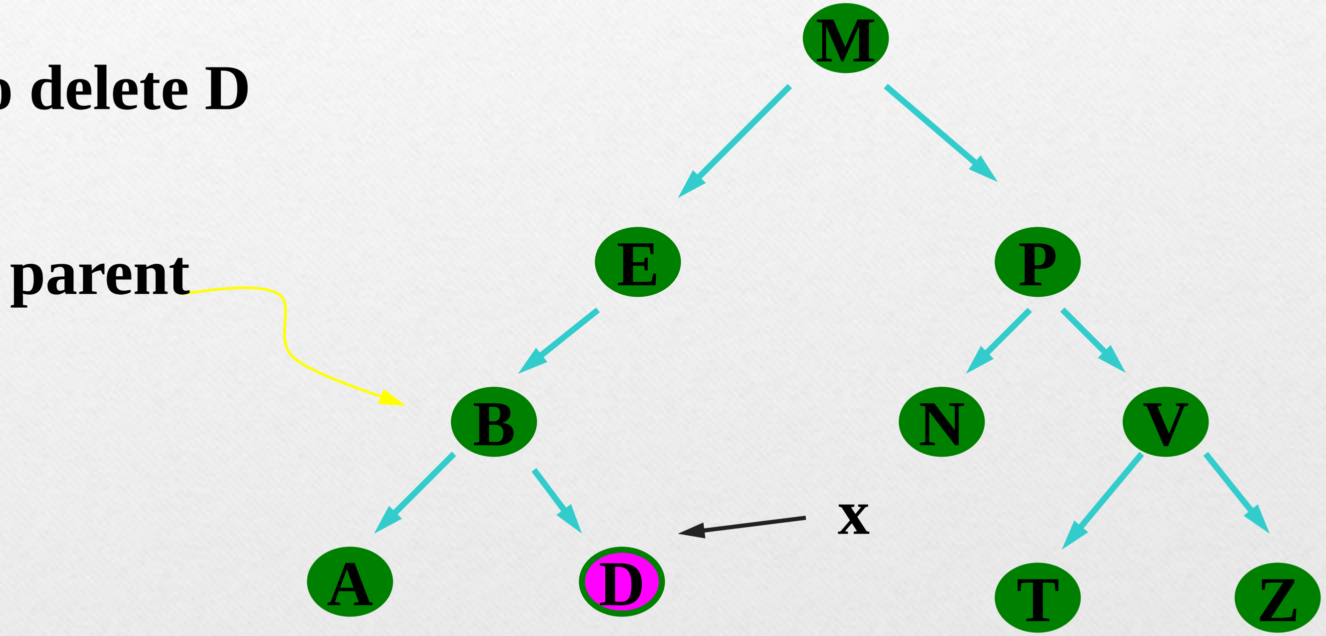
To delete D:

parent



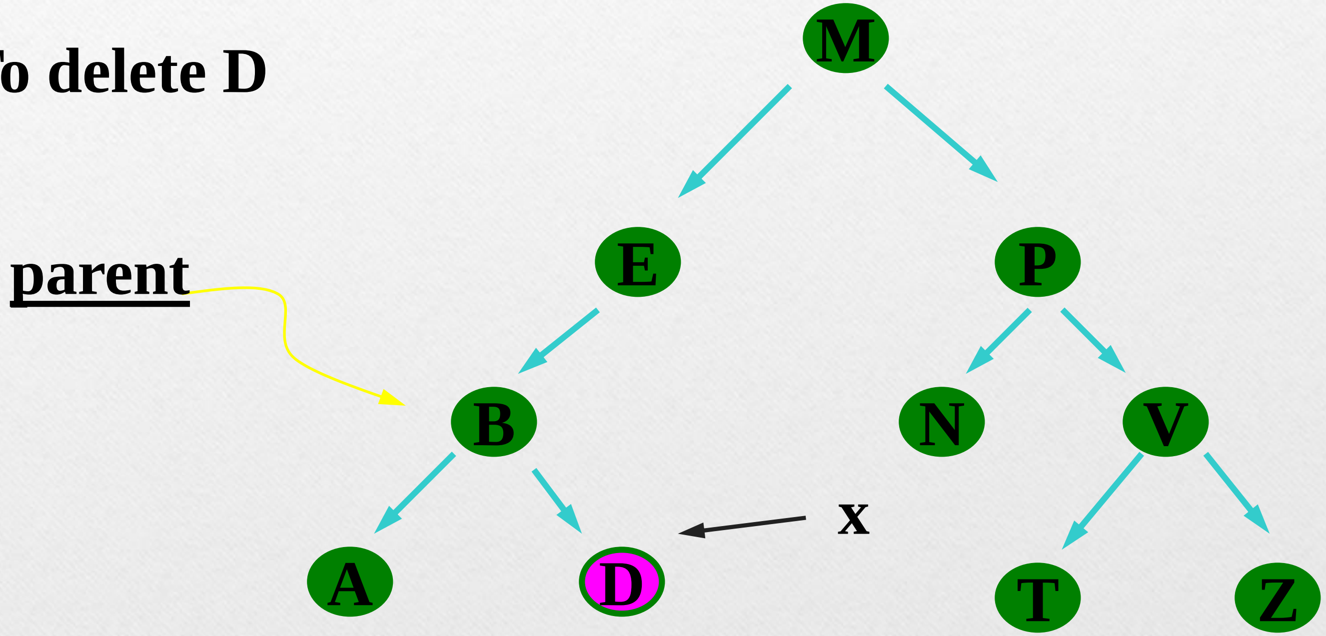
Delete Node: First Case

To delete D



Delete Node: First Case

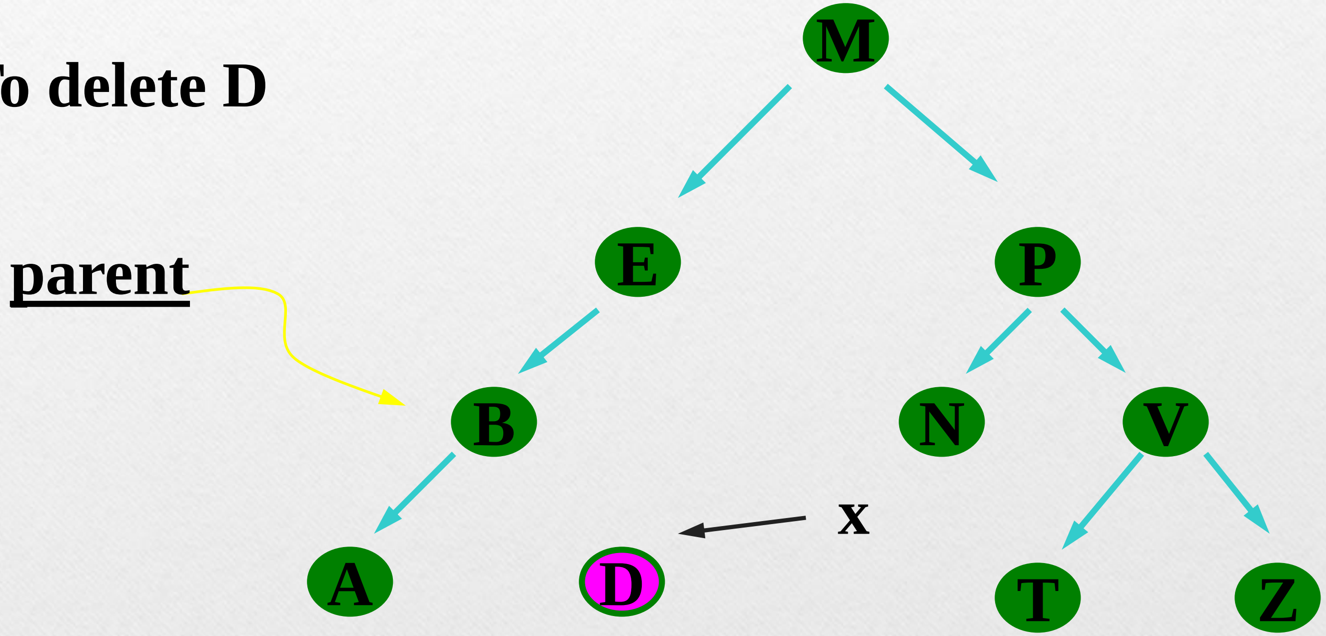
To delete D



Set the right child of parent as null

Delete Node: First Case

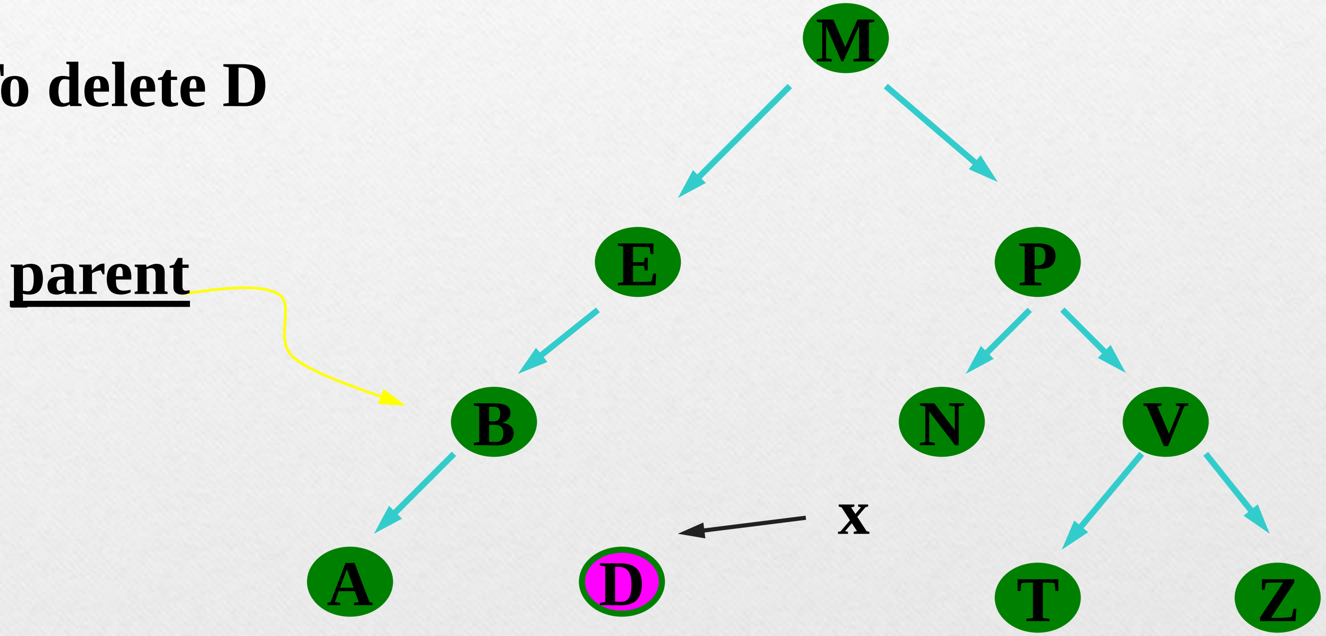
To delete D



Set the right child of <u>parent</u> as null

Delete Node: First Case

To delete D



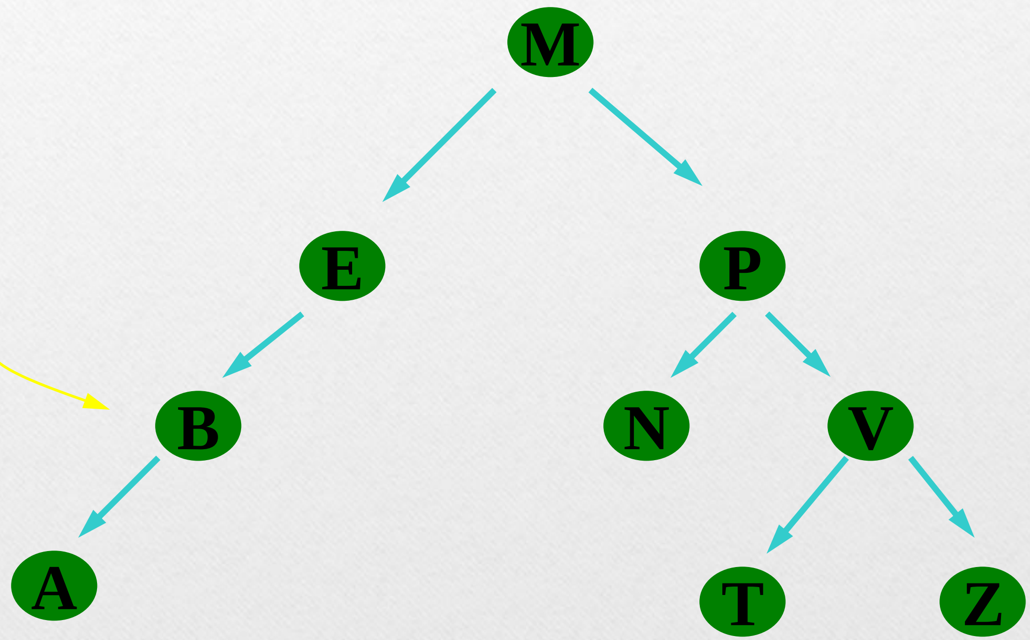
Set the right child of <u>parent</u> as null

Free x

Delete Node: First Case

To delete D

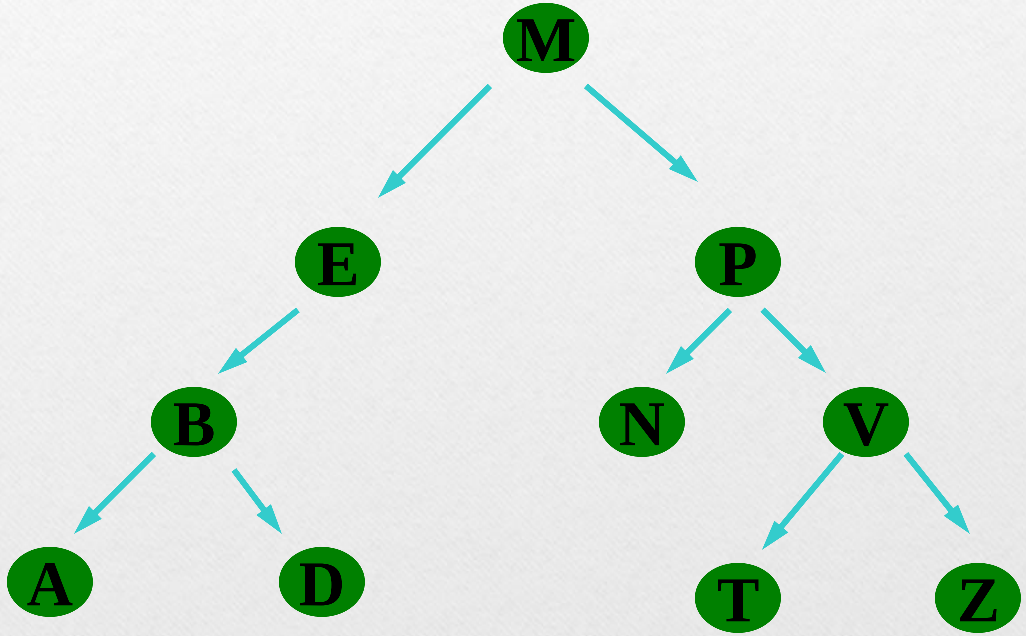
parent



Set the right child of **parent** as null
Free x

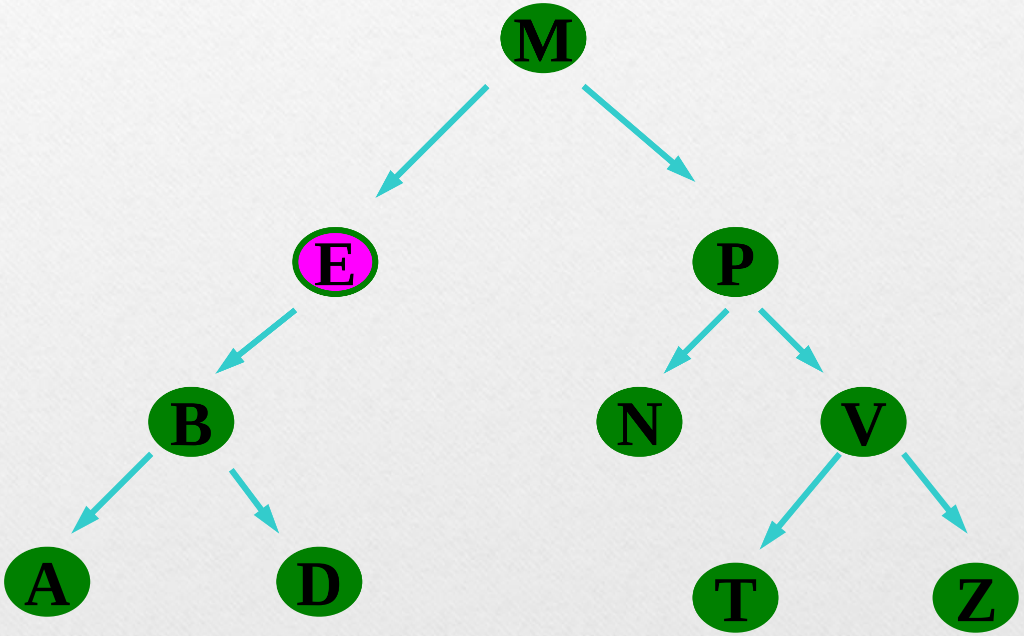
Delete Node: Second Case

To delete E



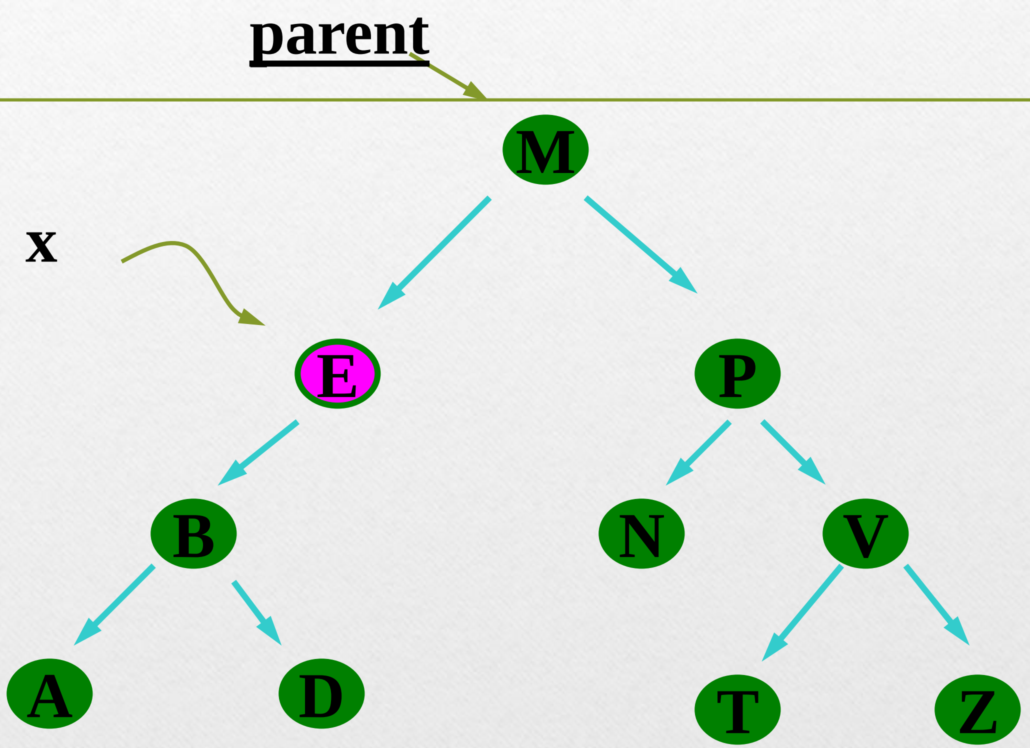
Delete Node: Second Case

To delete E

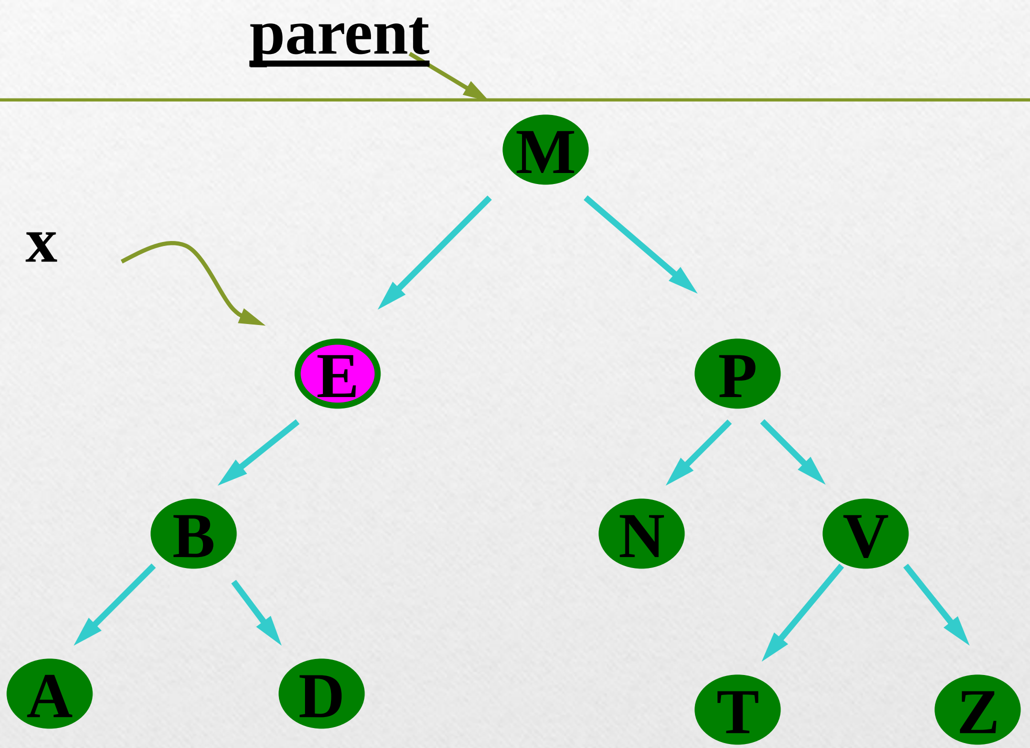


Delete Node: Second Case

To delete E



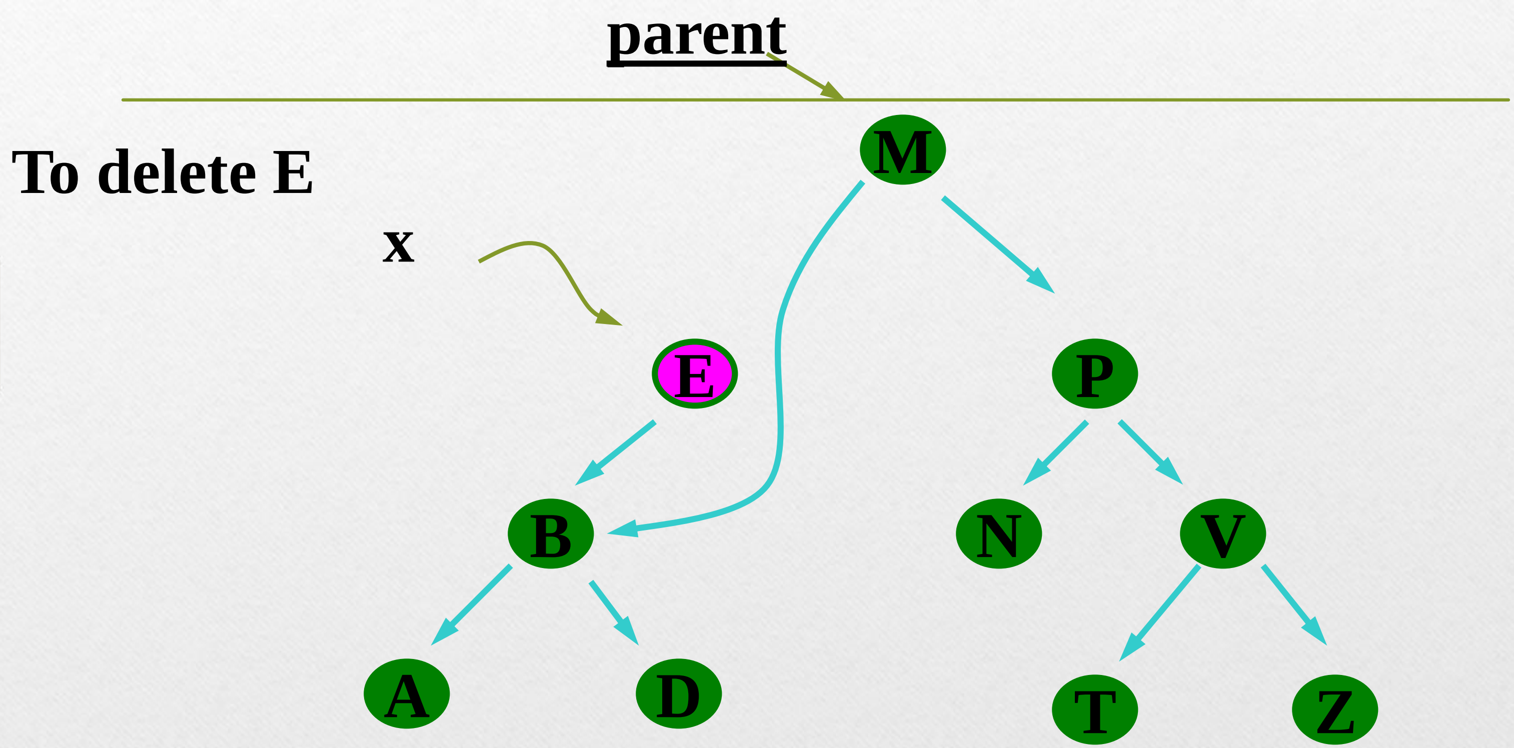
Delete Node: Second Case



To delete E

Set the Lchild (@ Rchild) of x as the Lchild (@ Rchild) of parent

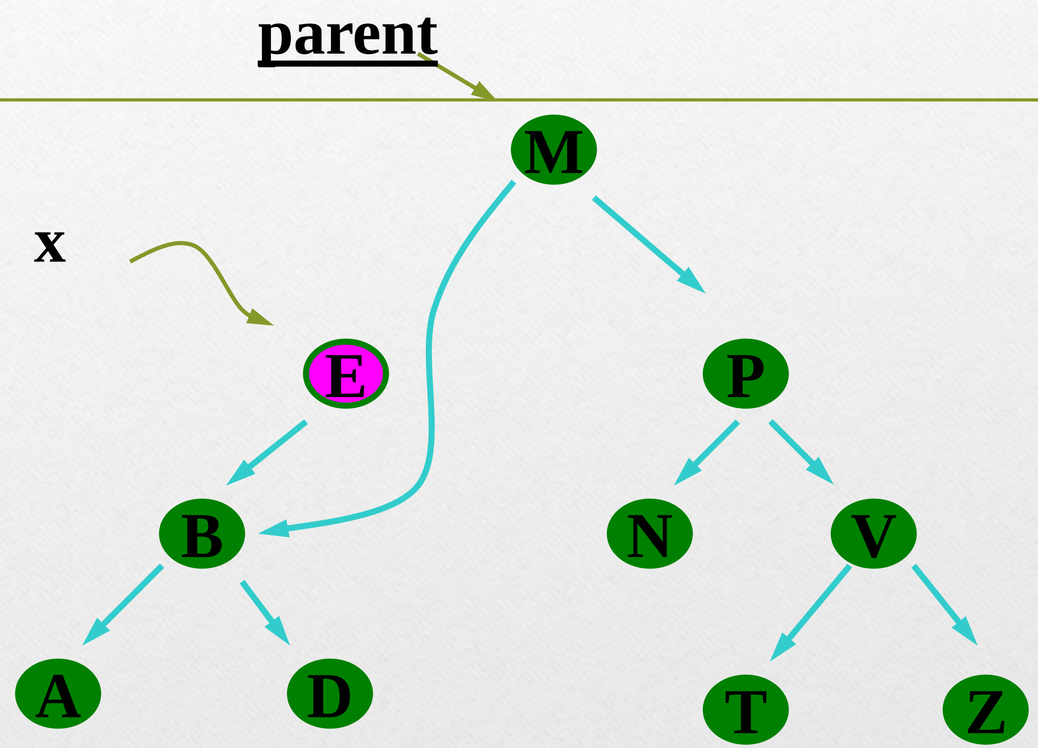
Delete Node: Second Case



Set the Lchild (@ Rchild) of x as the Lchild (@ Rchild) of parent

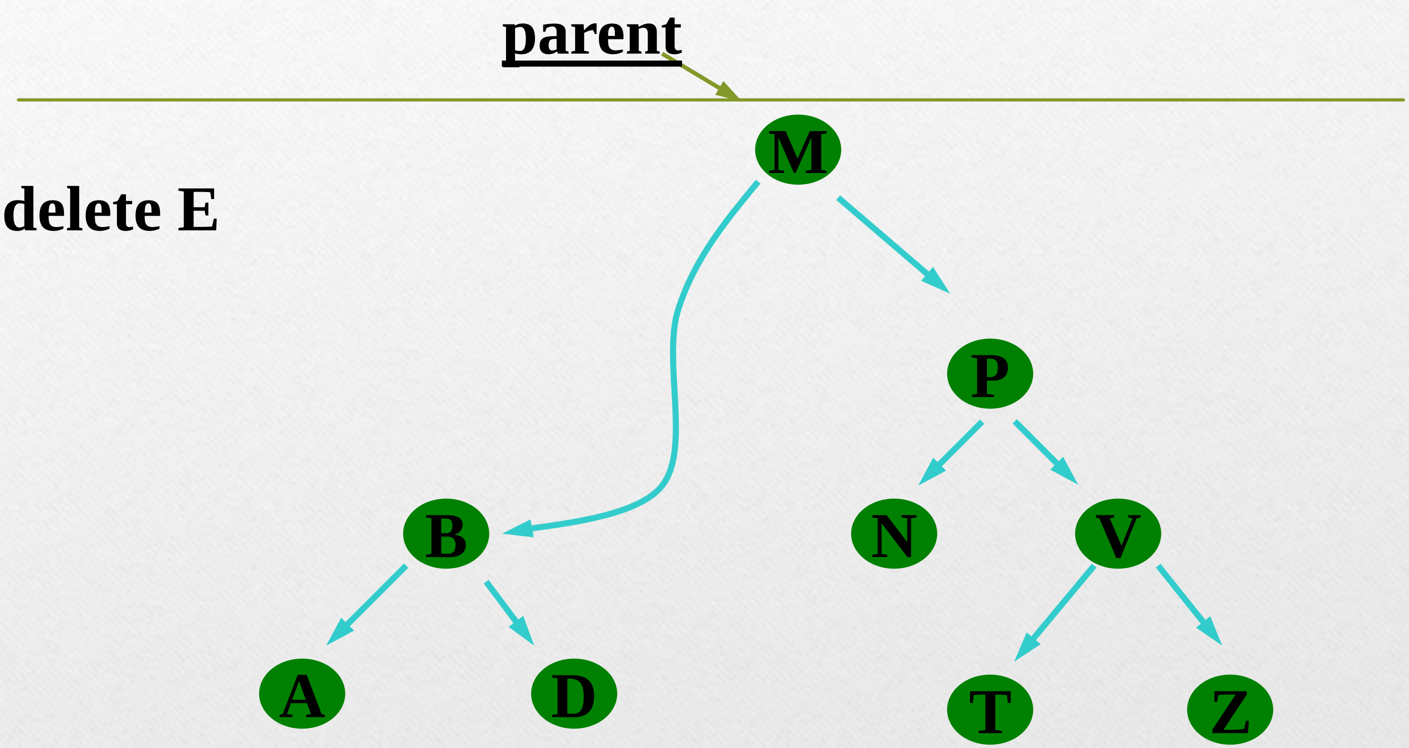
Delete Node: Second Case

To delete E



Set the Lchild (@ Rchild) of x as the Lchild (@ Rchild) of parent
Free x

Delete Node: Second Case



To delete E

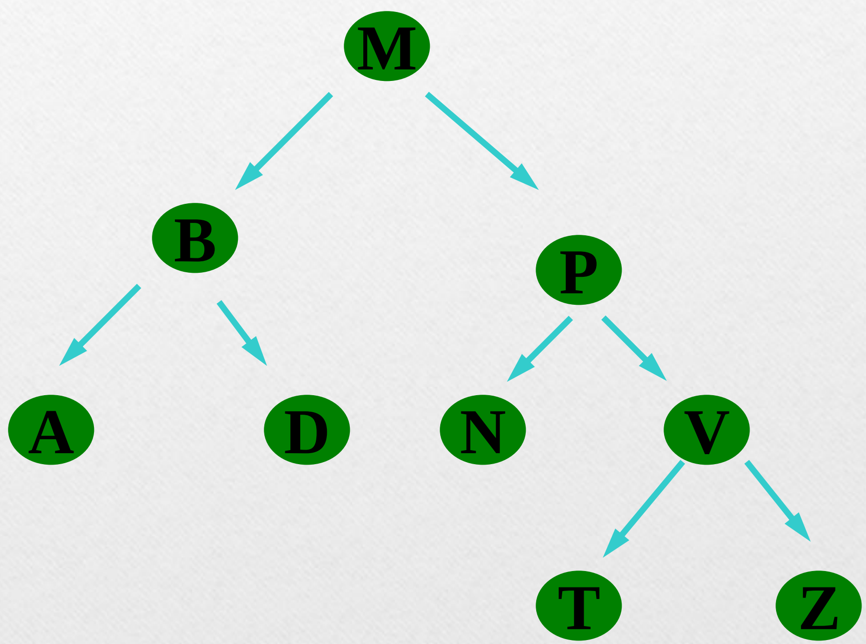
Set the Lchild (@ Rchild) of x as the Lchild (@ Rchild) of <u>parent</u>

Free x

Delete Node: Second Case

parent

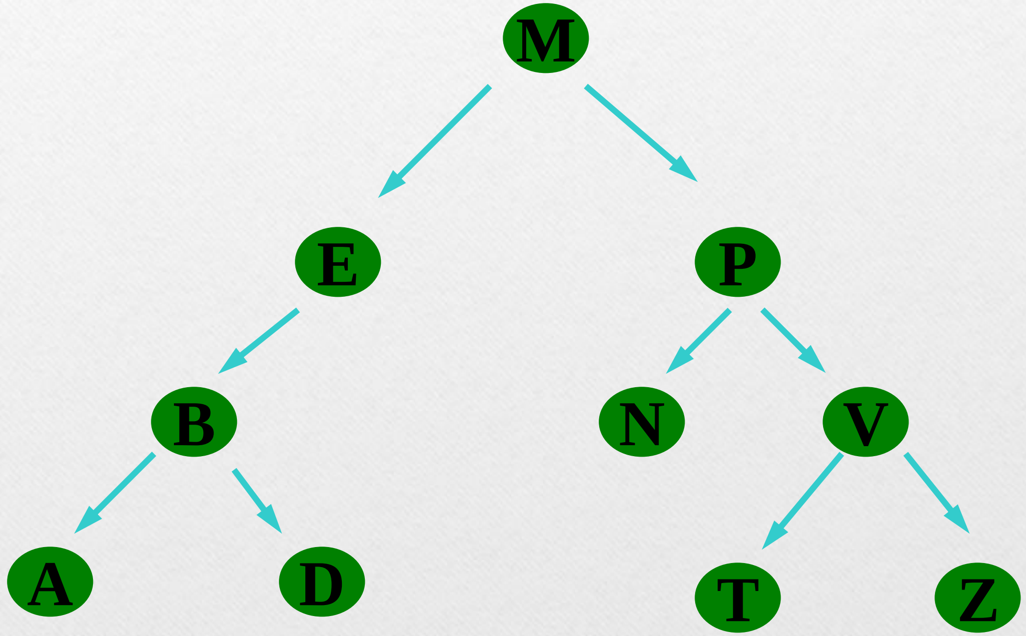
To delete E



Set the Lchild (@ Rchild) of x as the Lchild (@ Rchild) of parent
Free x

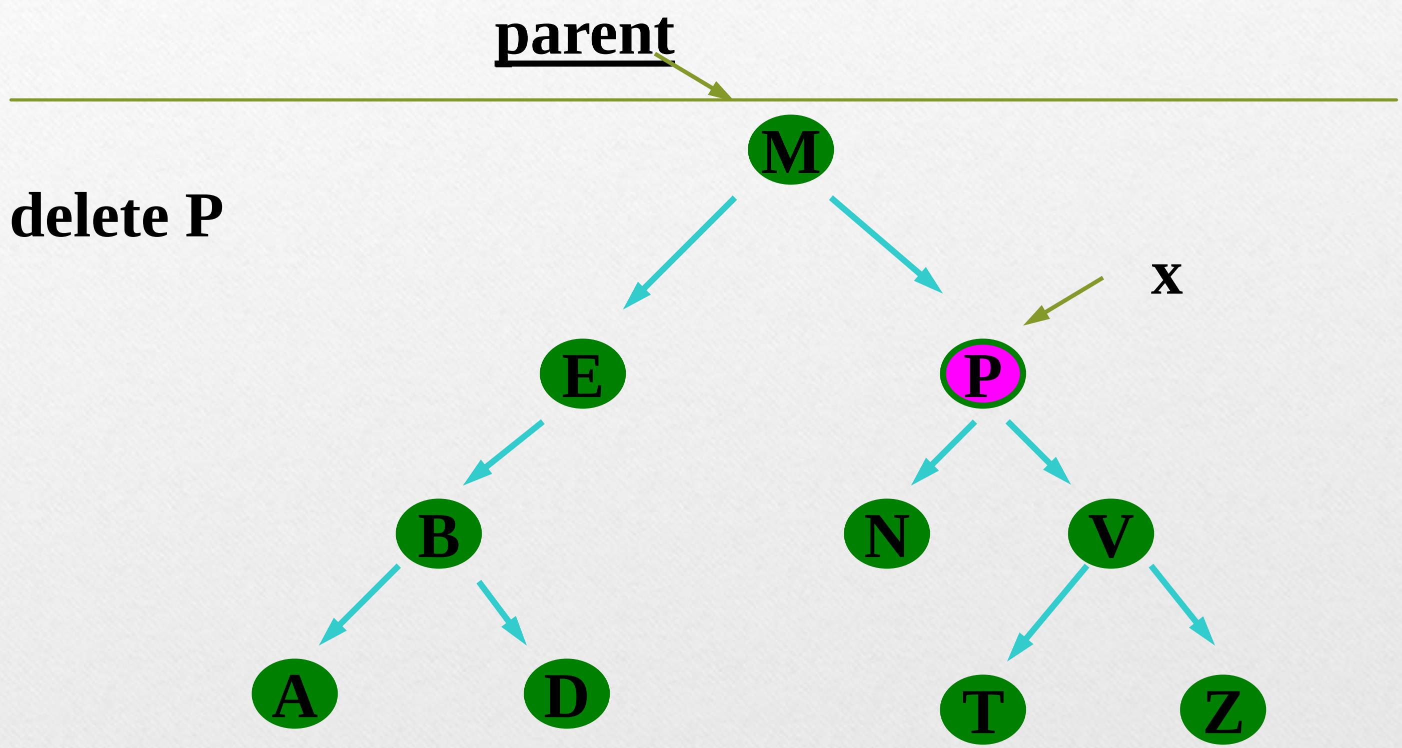
Delete Node: Third Case

To delete P



Delete Node: Third Case

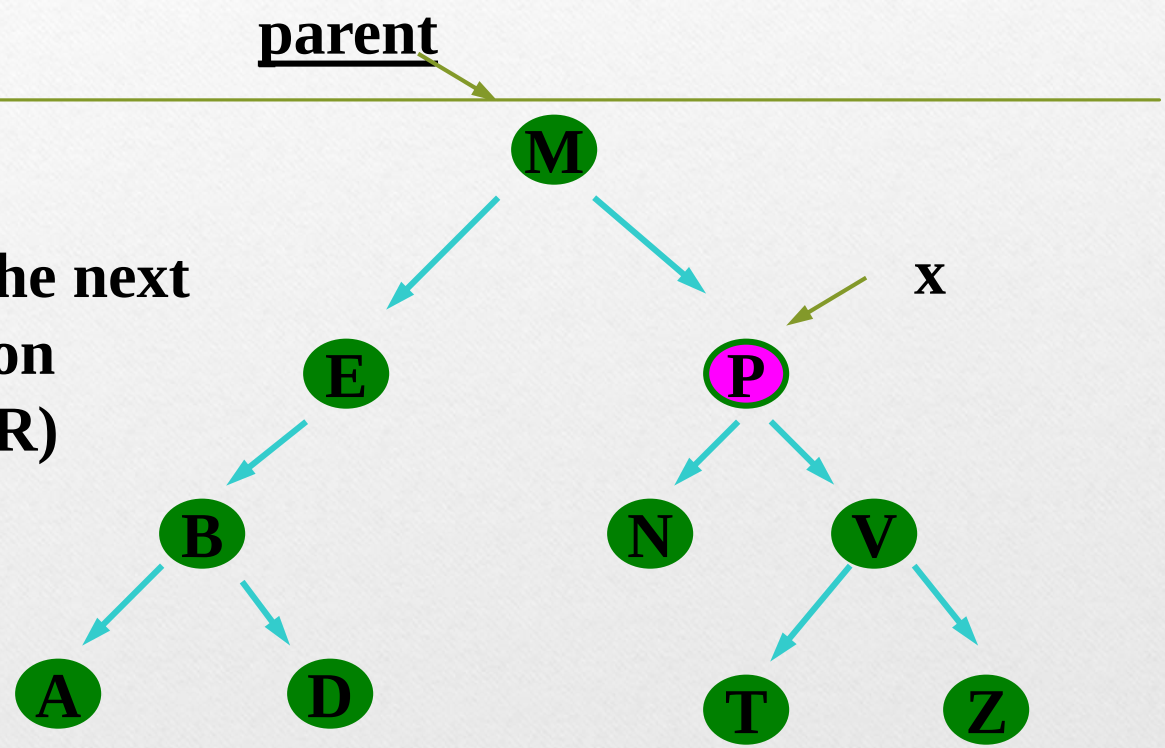
To delete P



Delete Node: Third Case

To delete P

Determine the next node based on Inorder(LNR)

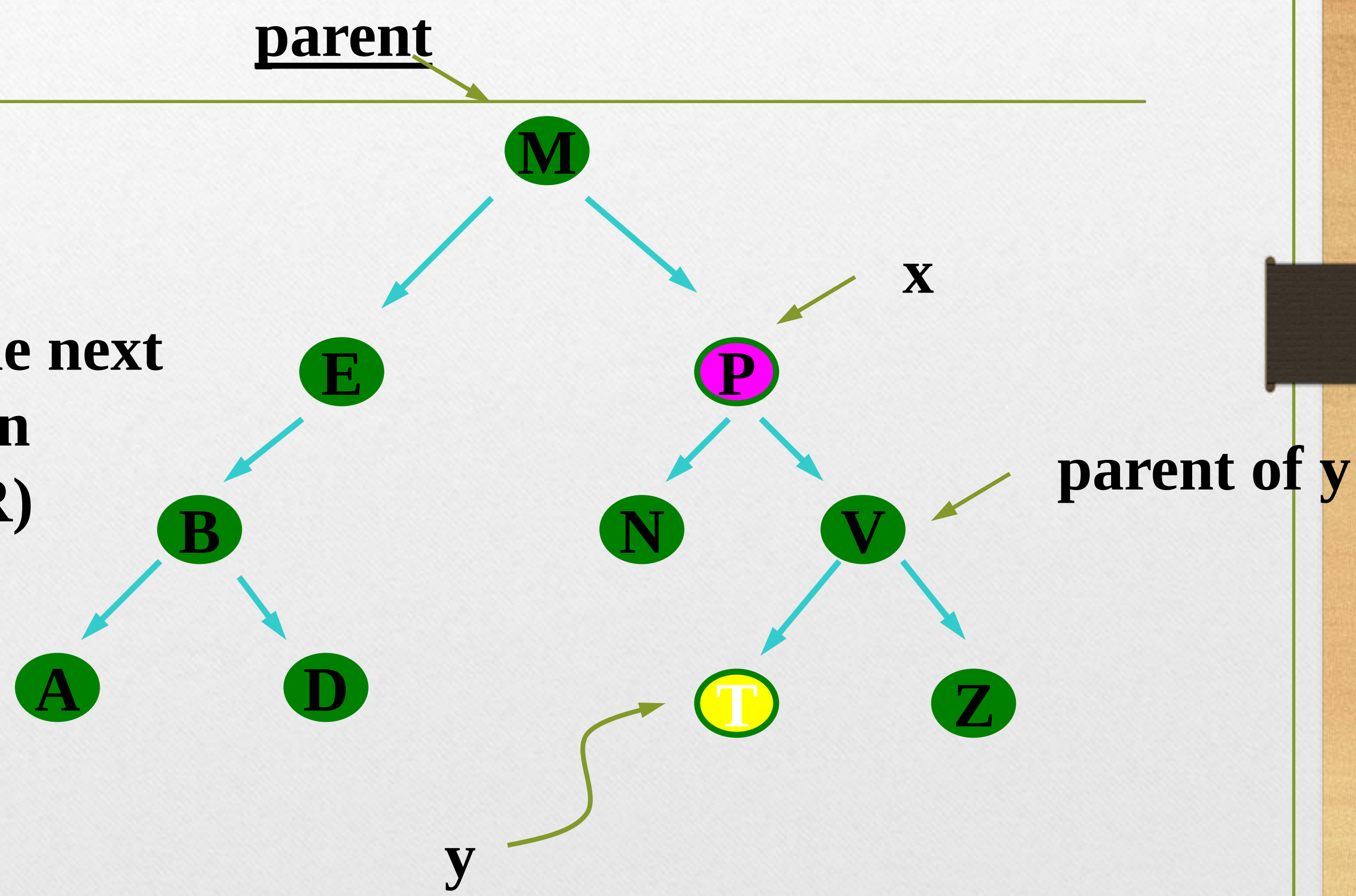


*Usually, the Inorder node has only one child or no children at all

Delete Node: Third Case

To delete P

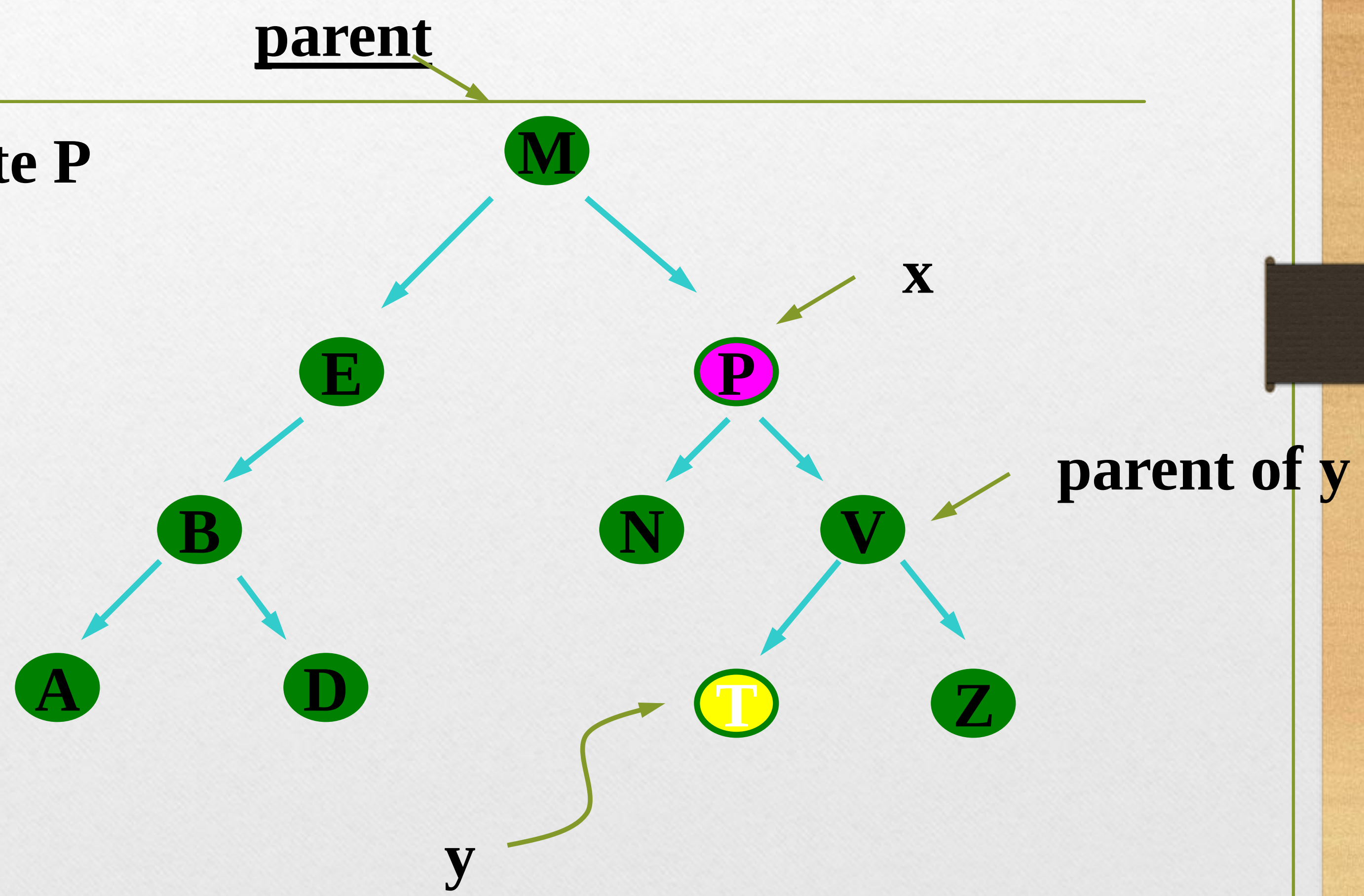
Determine the next node based on Inorder(LNR)



Delete Node: Third Case

To delete P

Copy

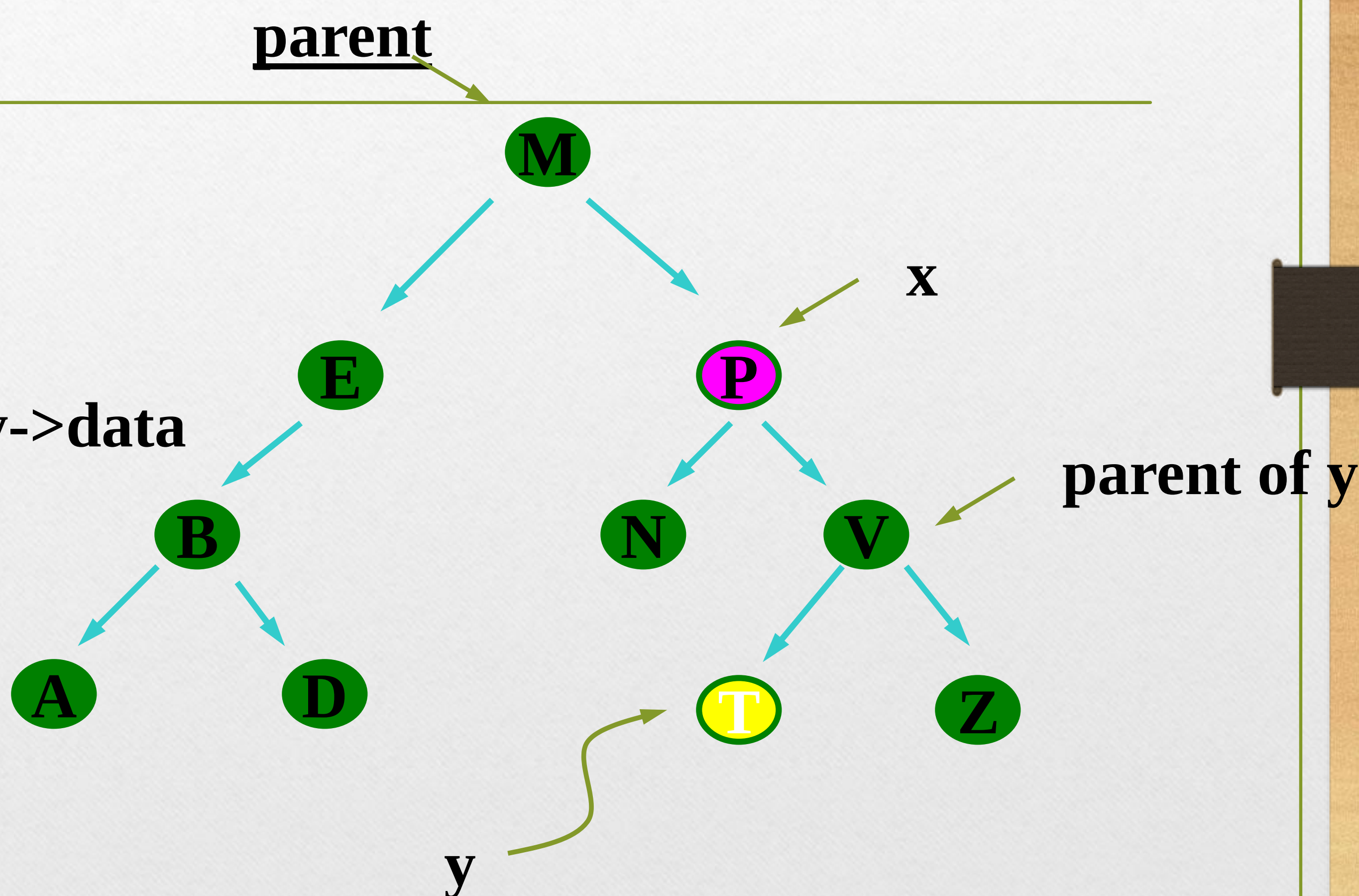


Delete Node: Third Case

To delete P

Copy

x->data = y->data

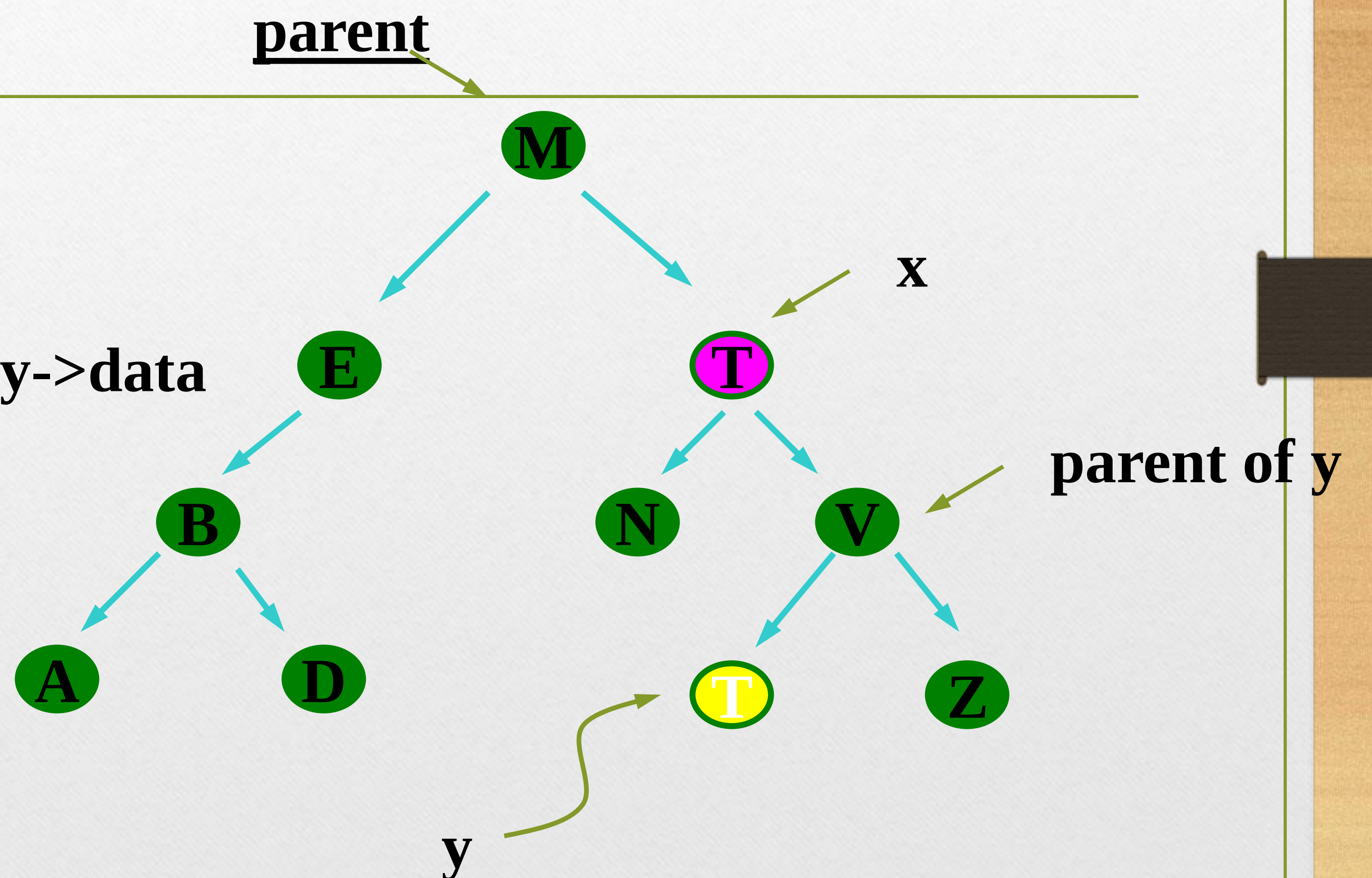


Delete Node: Third Case

To delete P

Copy

x->data = y->data



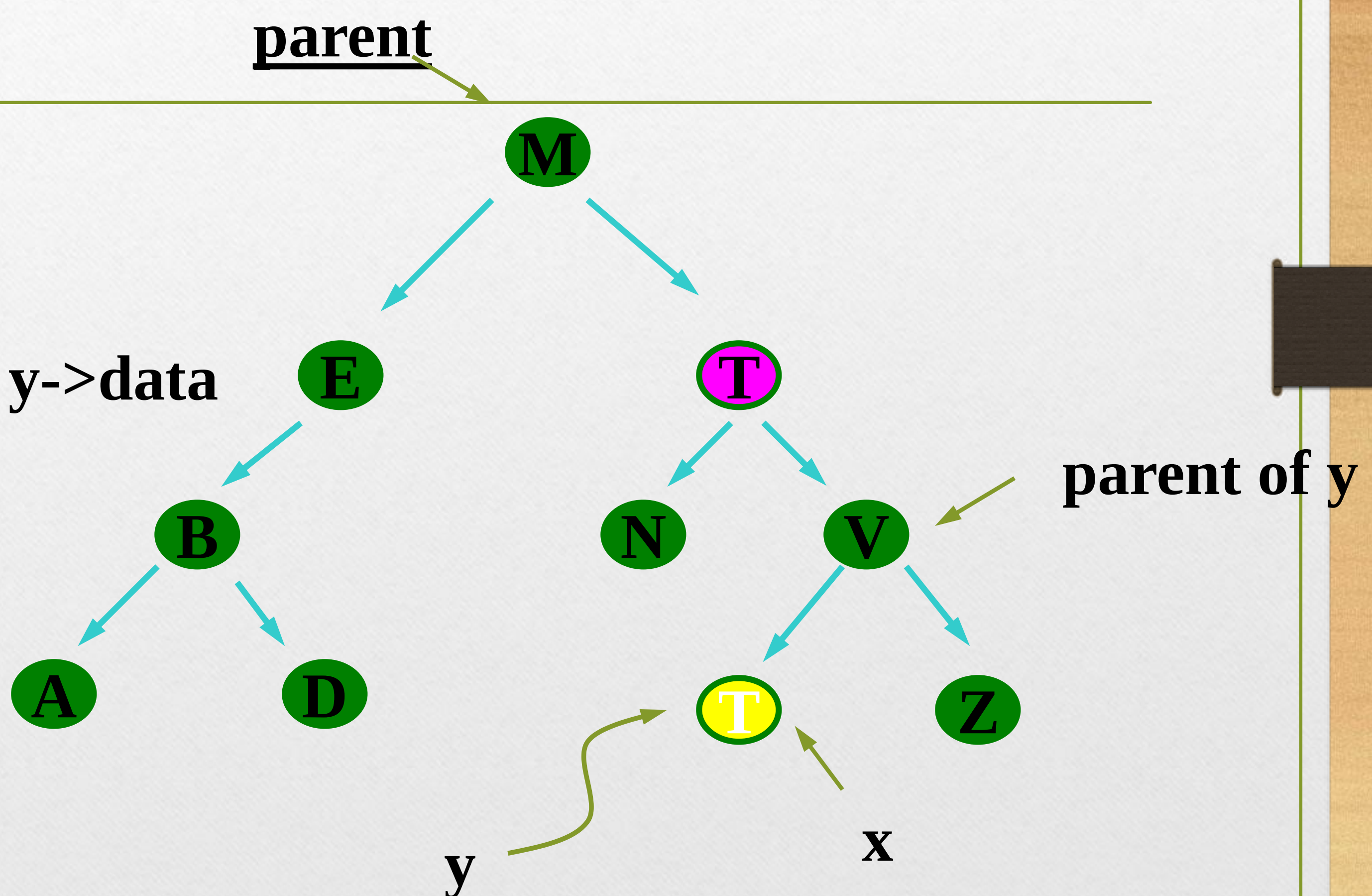
Delete Node: Third Case

To delete P

Copy

x->data = y->data

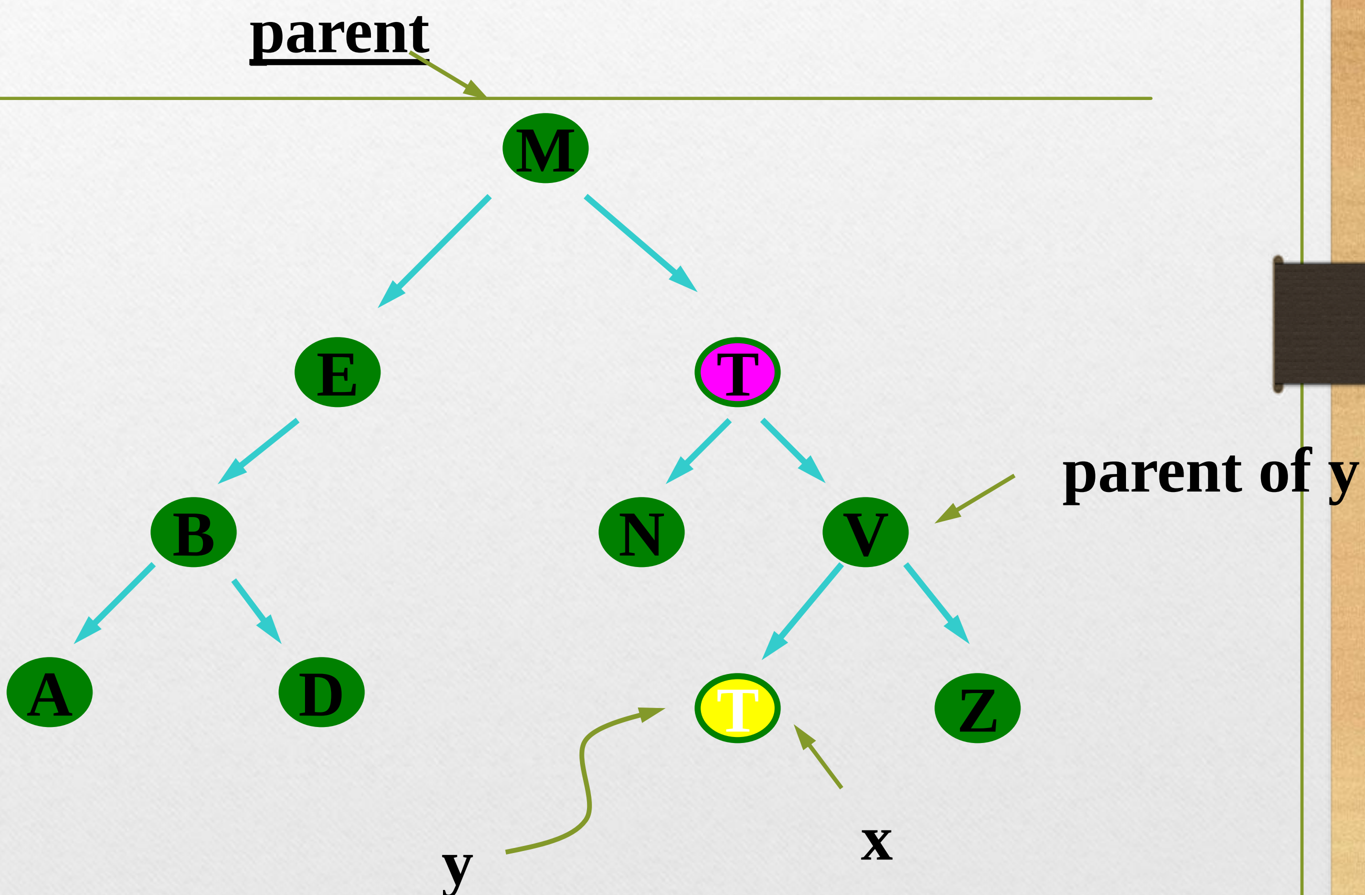
x = y



Delete Node: Third Case

To delete P

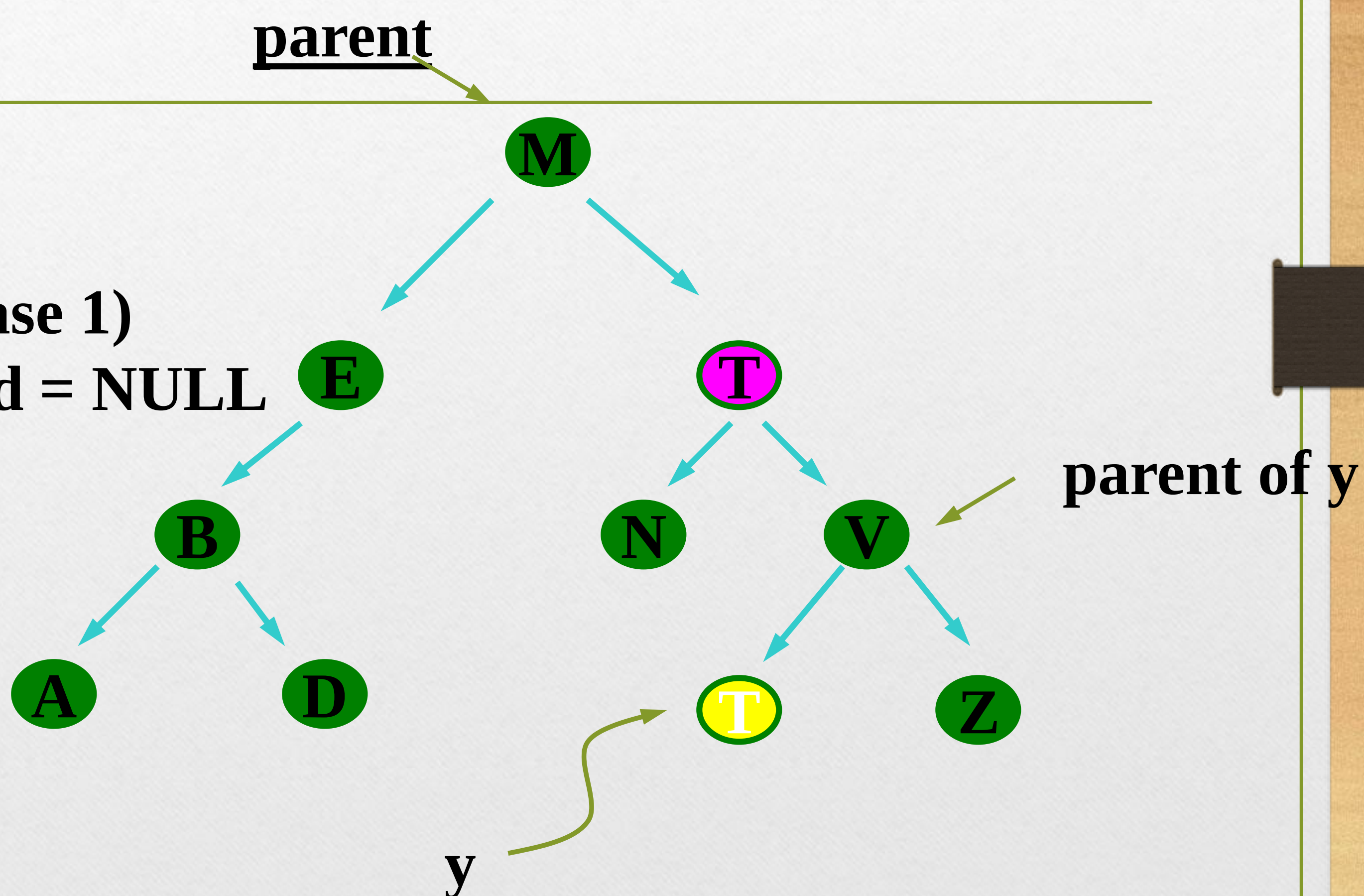
Delete T



Delete Node: Third Case

To delete P

Delete T (as in case 1)
parent_y->Lchild = NULL



Delete Node: Third Case

To delete P

Delete T