Data Structures and Object Oriented Programming

Lecture 15

Dr. Naveed Anwar Bhatti

Webpage: naveedanwarbhatti.github.io

Object-Oriented Programming in C++

Binary Search Trees

Deletion

BST – Deletion

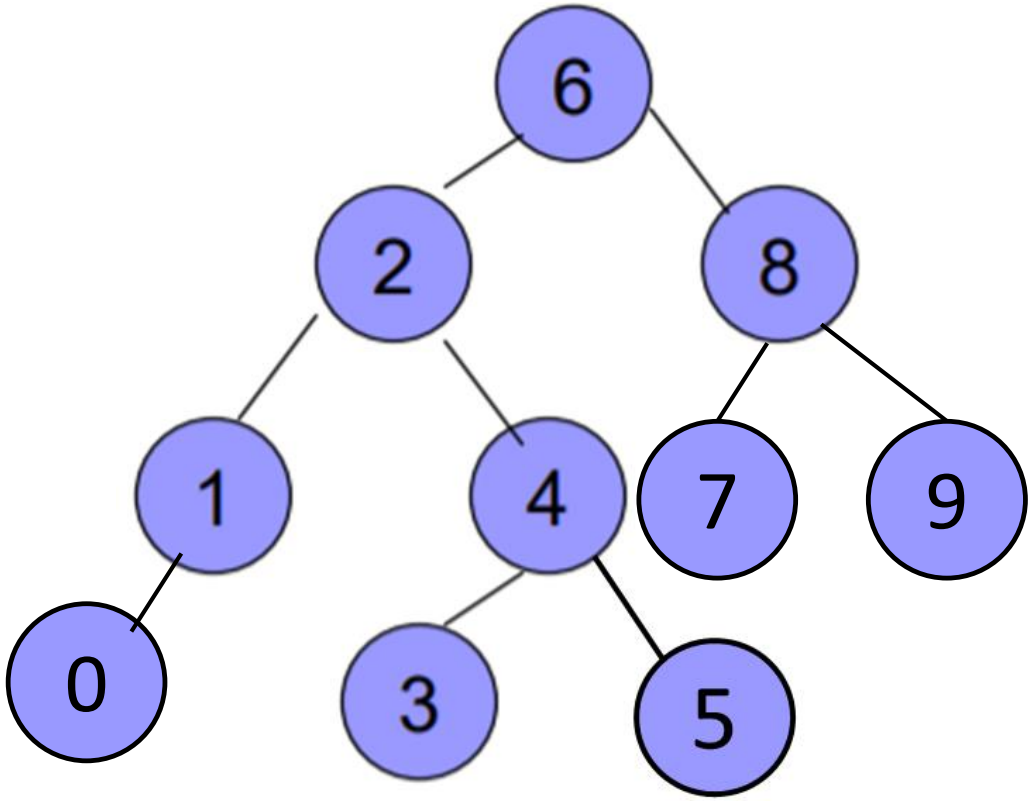
Deletion of a node **N** from the tree depends primarily on the number of children of node **N**.

There are three cases:

- Case 1: No Children
- Case 2: One Child
- Case 3: Two Children

BST – Deletion

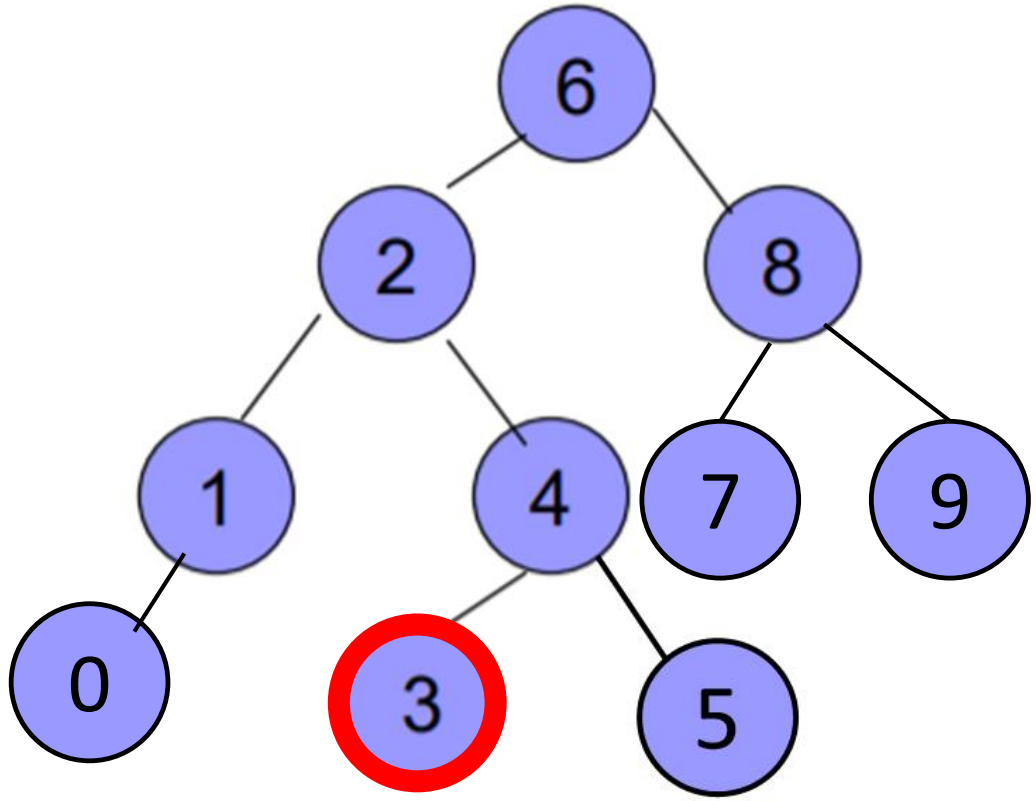
Case 1: No Children



BST – Deletion

Case 1: No Children

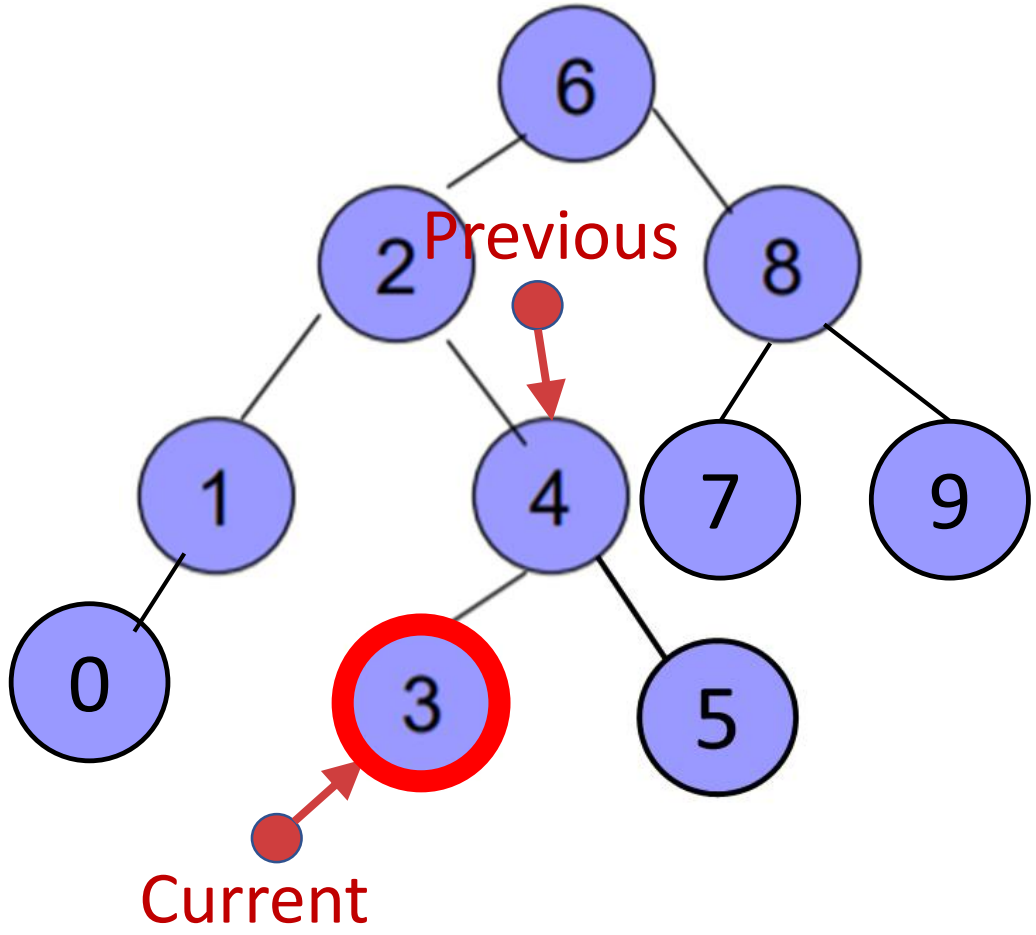
Delete ‘3’



BST – Deletion

Case 1: No Children

Delete ‘3’

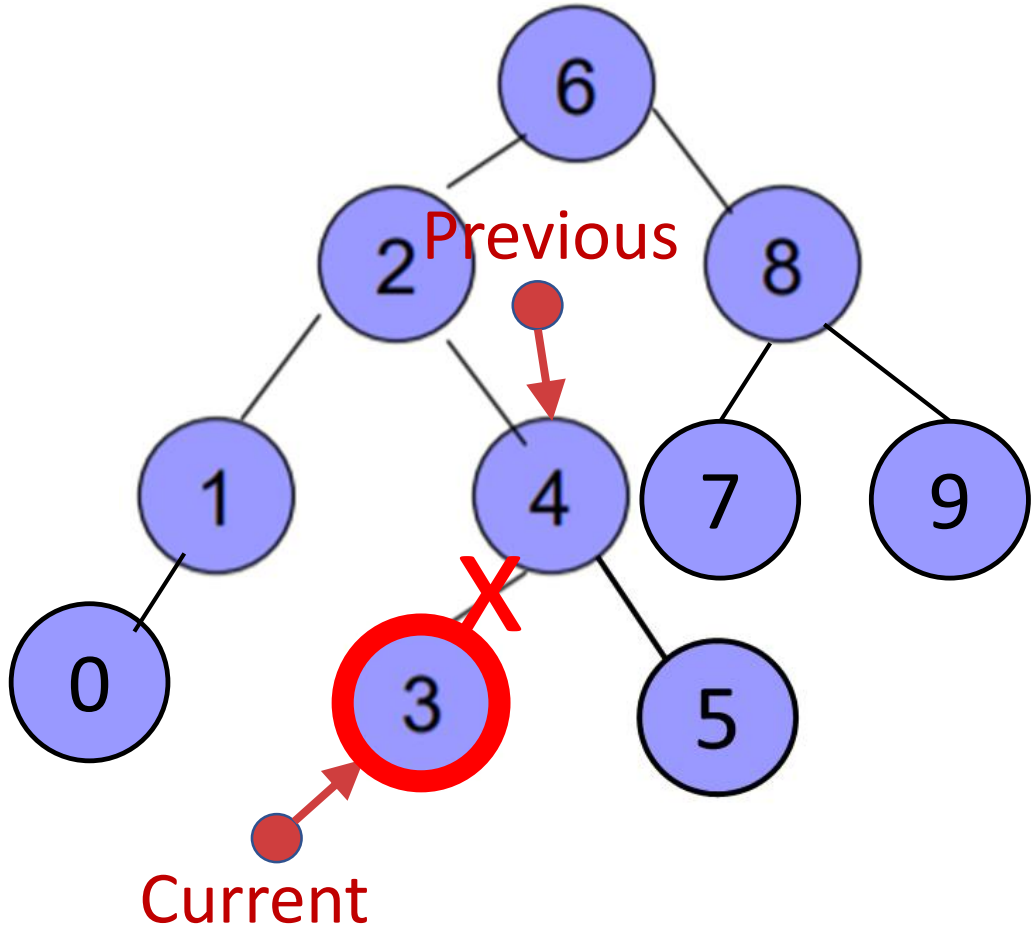


BST – Deletion

Case 1: No Children

Delete ‘3’

```
if (Previous->right == Current)
        Previous->right = NULL
else
        Previous->left = NULL
```

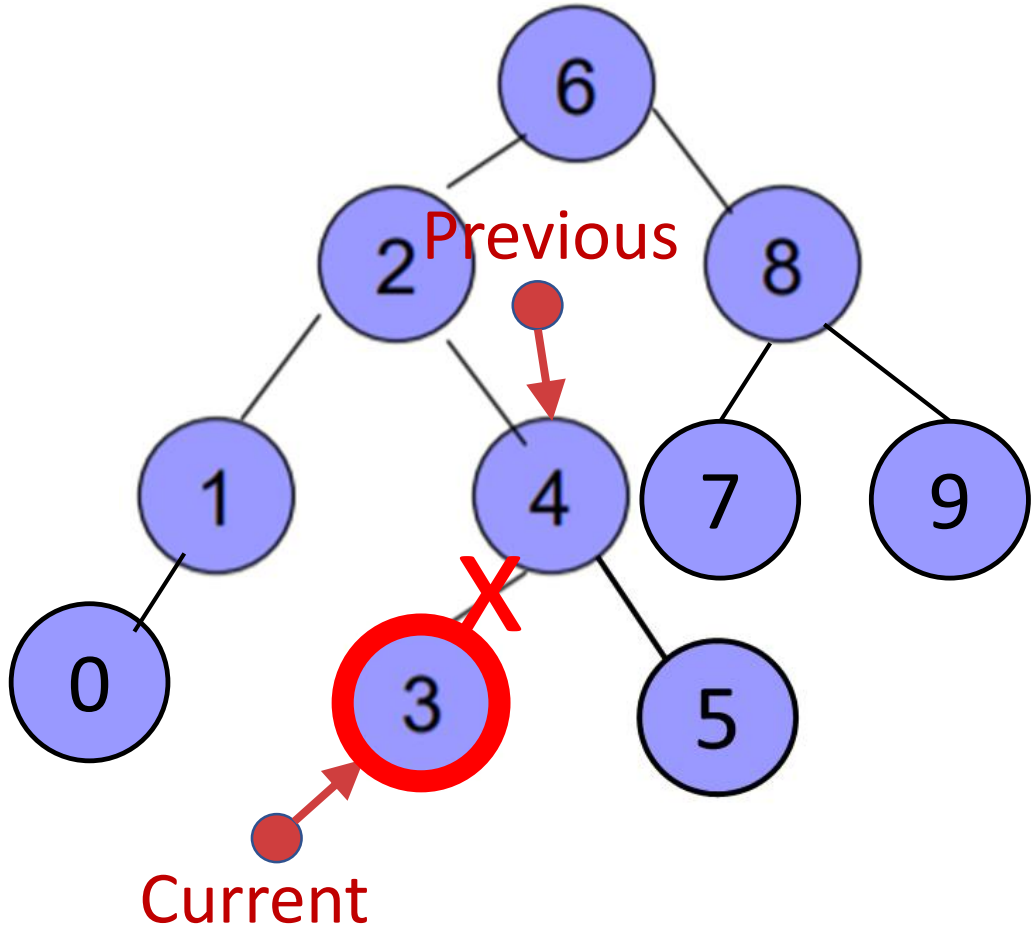


BST – Deletion

Case 1: No Children

Delete ‘3’

```
if (Previous->right == Current)
        Previous->right = NULL
else
        Previous->left = NULL
```

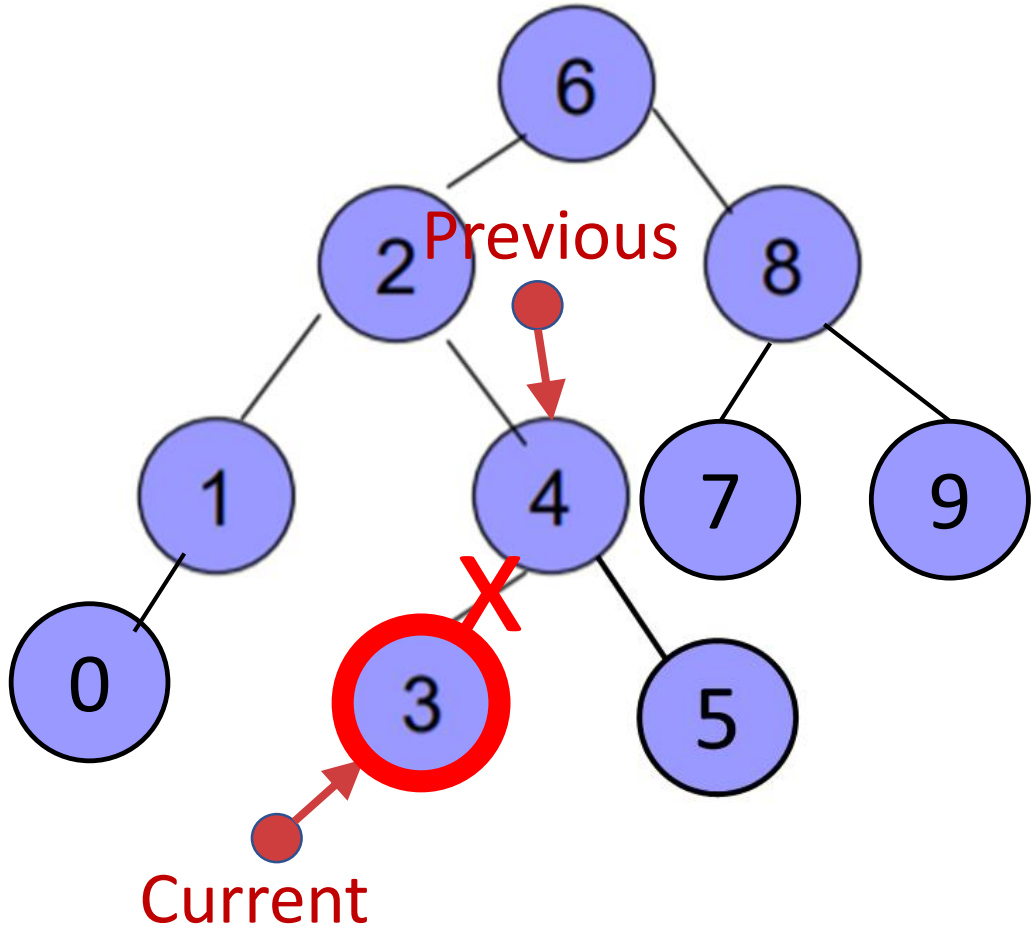


BST – Deletion

Case 1: No Children

Delete ‘3’

```
if (Previous->right == Current)
        Previous->right = NULL
else
        Previous->left = NULL
```

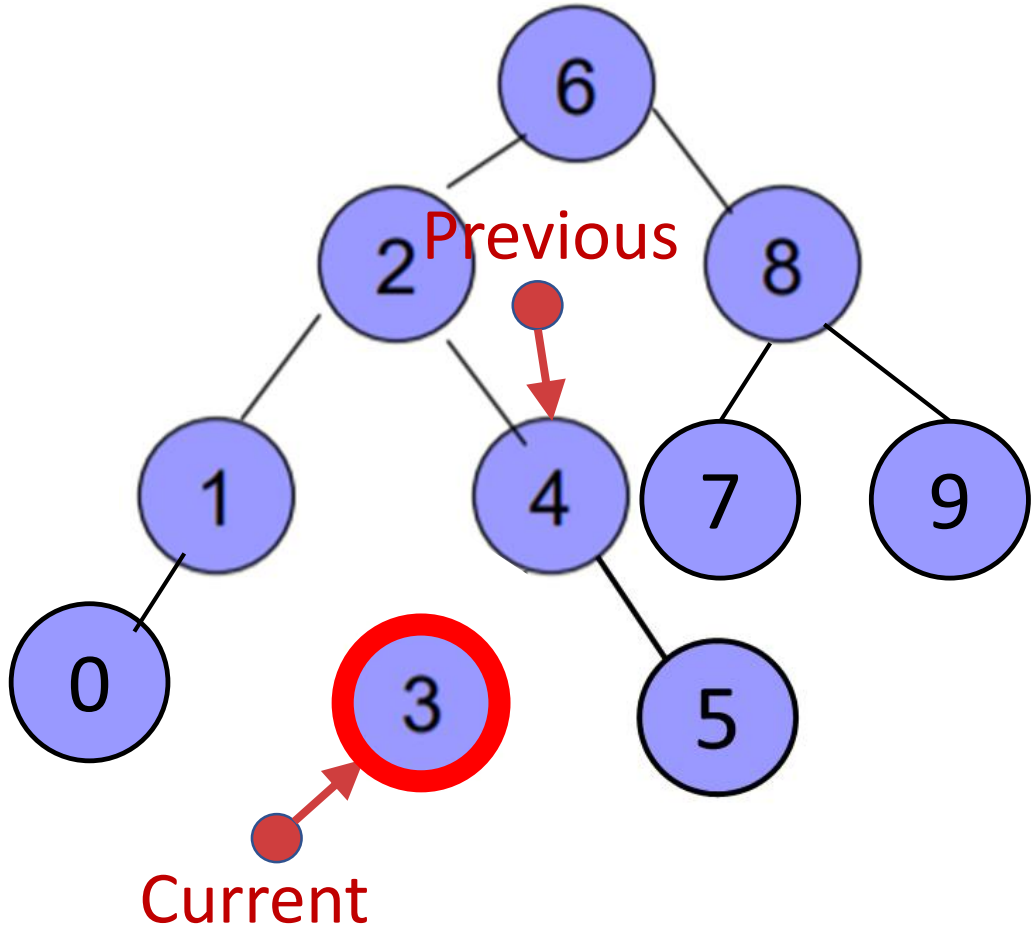


BST – Deletion

Case 1: No Children

Delete ‘0’

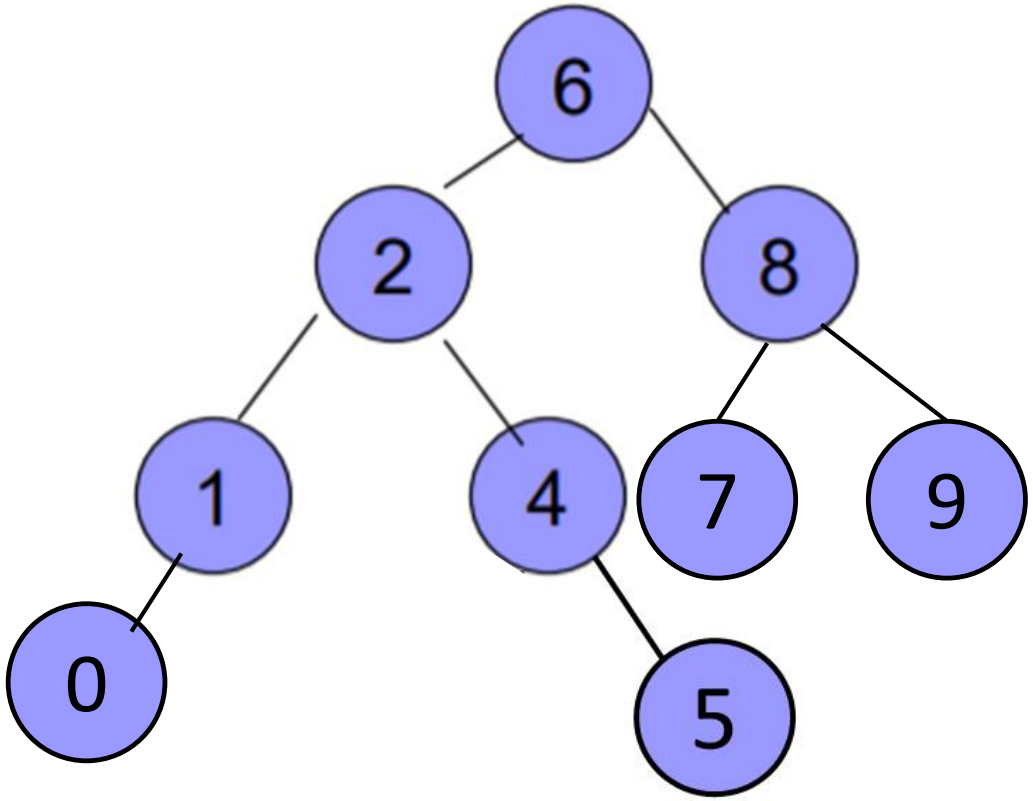
Delete Current



BST – Deletion

Case 1: No Children

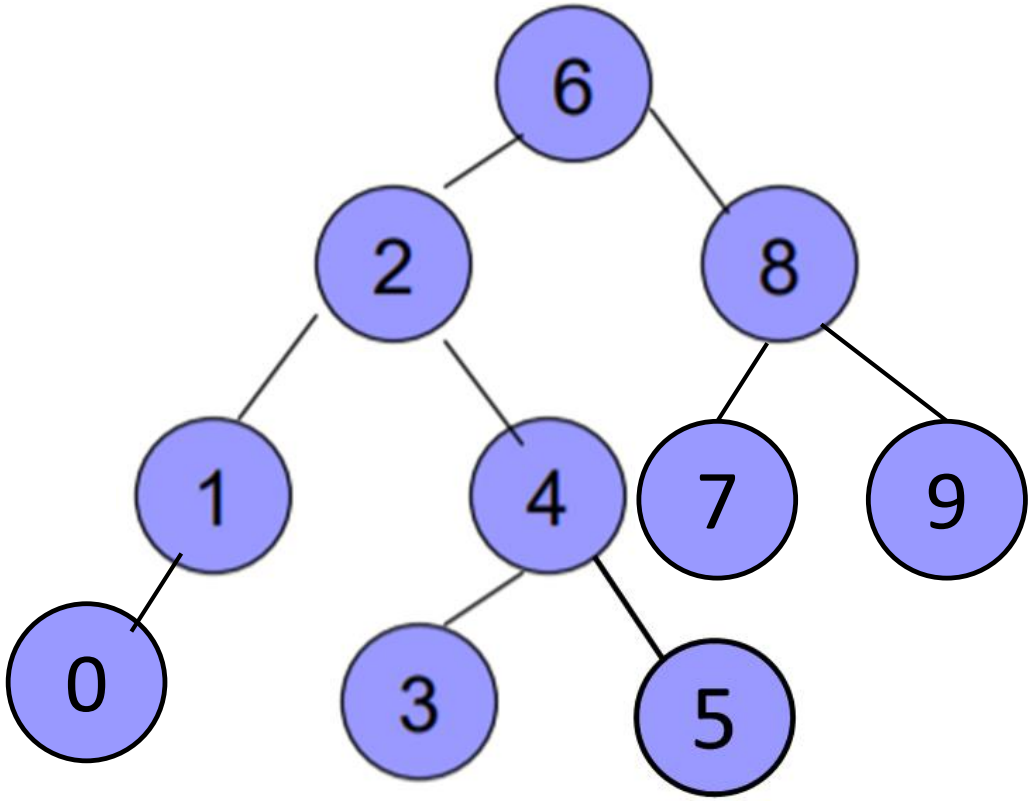
Delete ‘0’



BST – Deletion

Case 2: One child

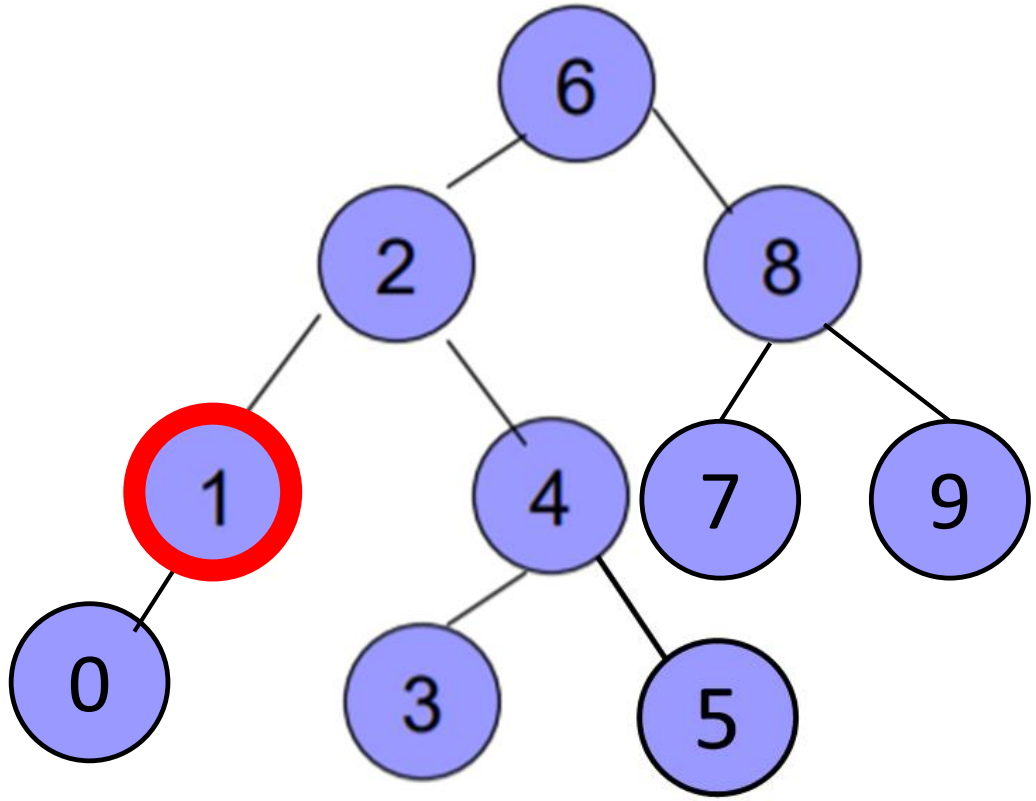
Delete ‘1’



BST – Deletion

Case 2: One child

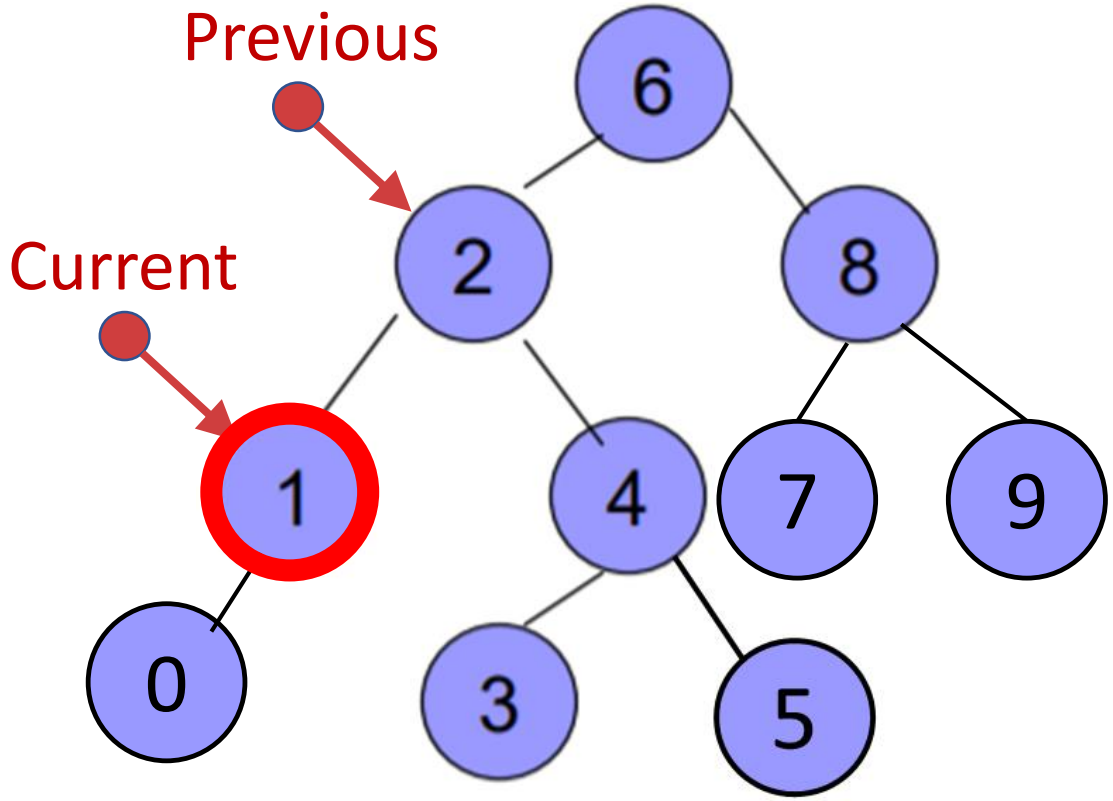
Delete ‘1’



BST – Deletion

Case 2: One child

Delete ‘1’



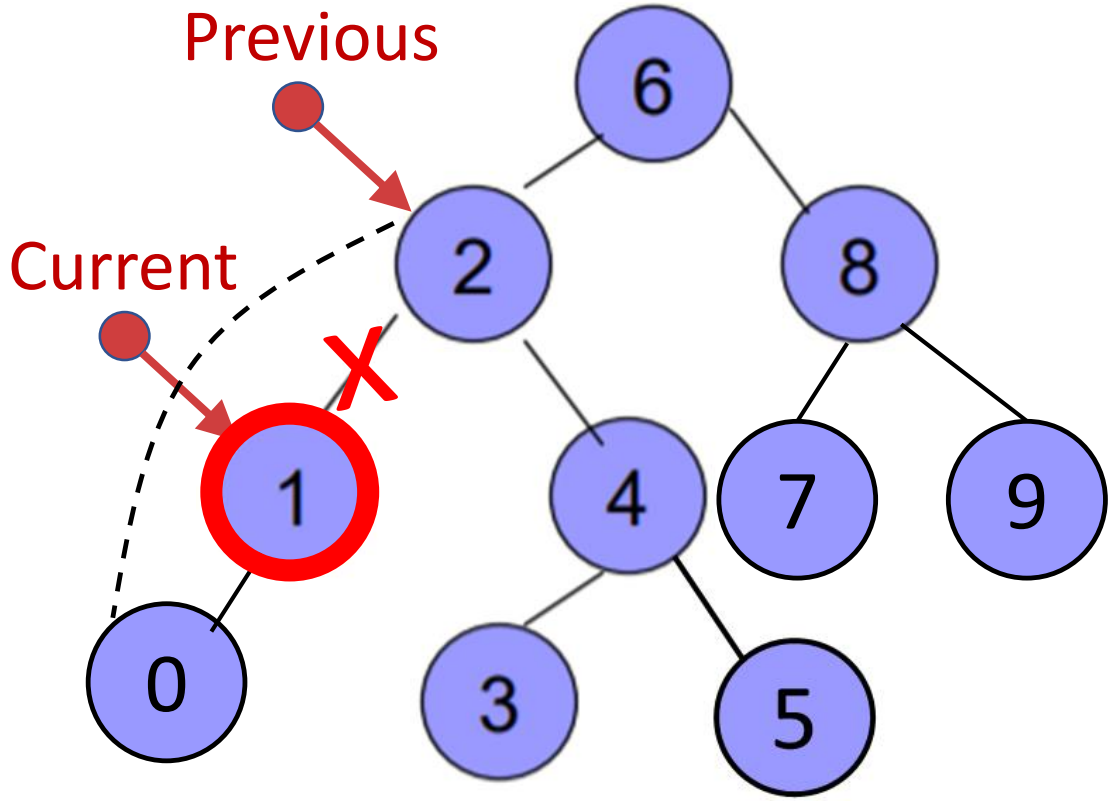
BST – Deletion

Case 2: One child

Delete ‘1’

```
if (Previous->right == Current)
        If(Current->right!=NULL)
                Previous->right= Current->right;
        else
                Previous->right= Current->left;

else
        If(Current->right!=NULL)
                Previous->left= Current->right;
        else
                Previous->left= Current->left;
```



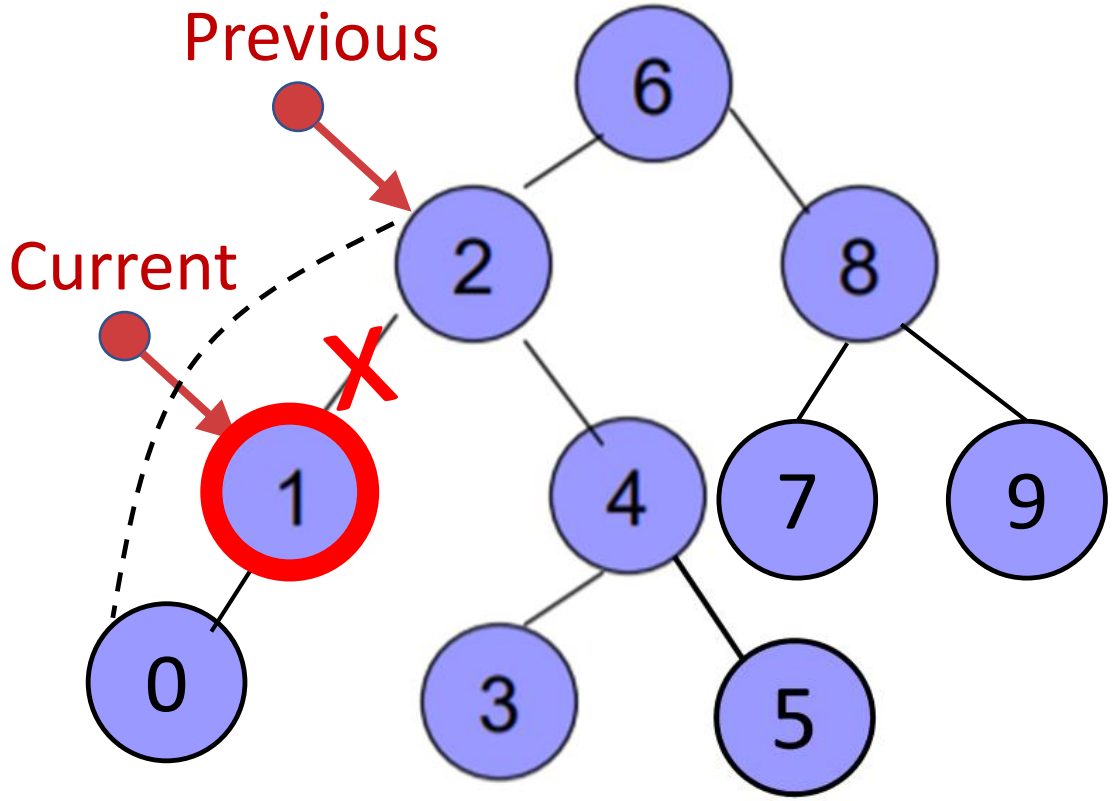
BST – Deletion

Case 2: One child

Delete ‘1’

```
if (Previous->right == Current)
        If(Current->right!=NULL)
                Previous->right= Current->right;
        else
                Previous->right= Current->left;

else
        If(Current->right!=NULL)
                Previous->left= Current->right;
        else
                Previous->left= Current->left;
```



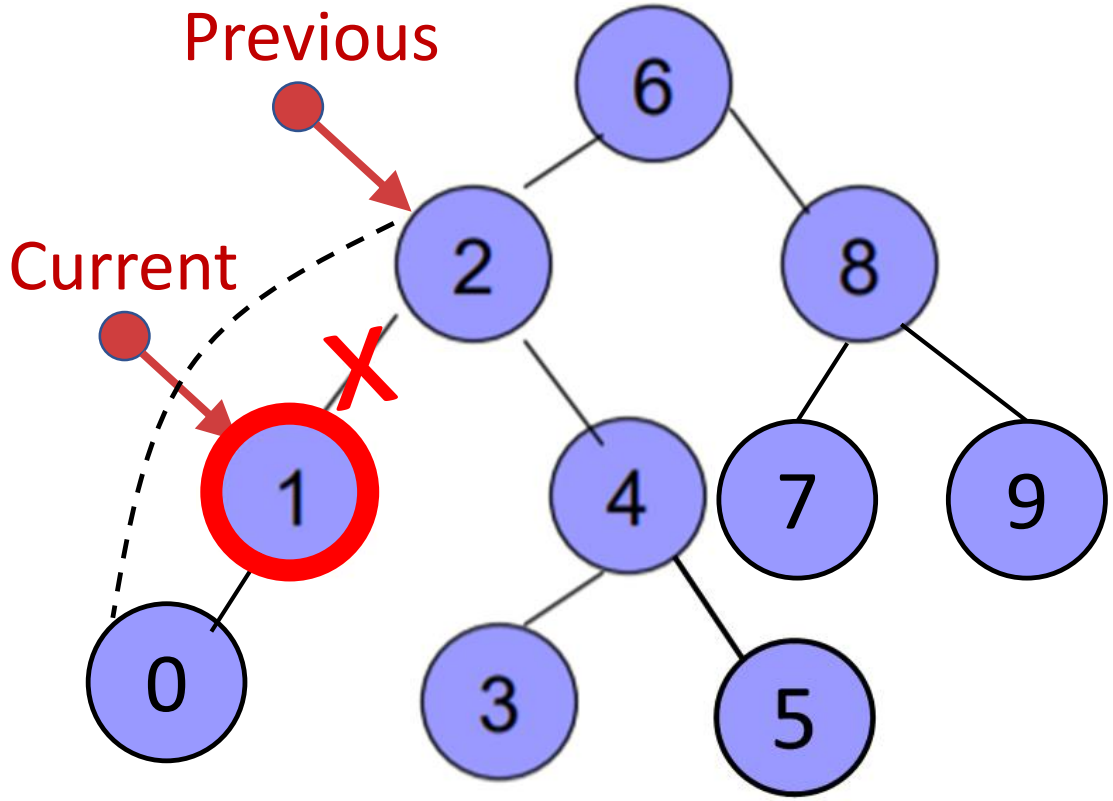
BST – Deletion

Case 2: One child

Delete ‘1’

```
if (Previous->right == Current)
        If(Current->right!=NULL)
                Previous->right= Current->right;
        else
                Previous->right= Current->left;


else
        If(Current->right!=NULL)
                Previous->left= Current->right;
        else
                Previous->left= Current->left;
```



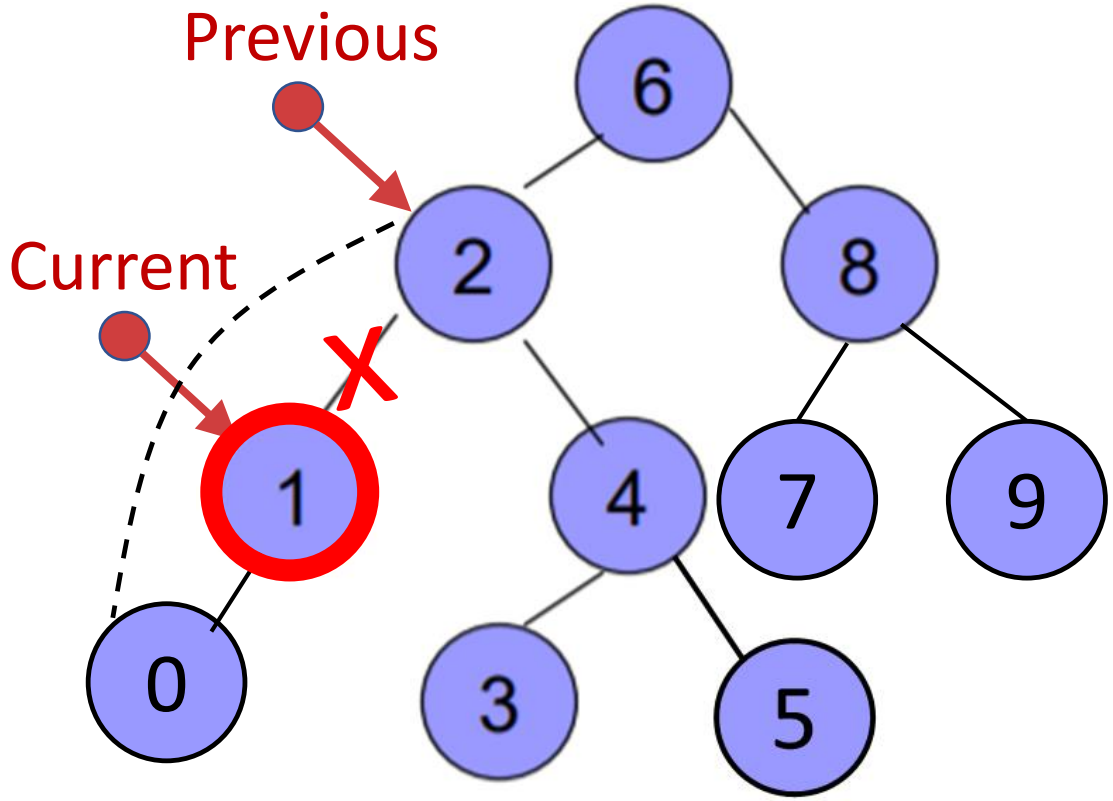
BST – Deletion

Case 2: One child

Delete ‘1’

```
if (Previous->right == Current)
        If(Current->right!=NULL)
                Previous->right= Current->right;
        else
                Previous->right= Current->left;

else
        If(Current->right!=NULL)
                Previous->left= Current->right;
        else
                Previous->left= Current->left;
```



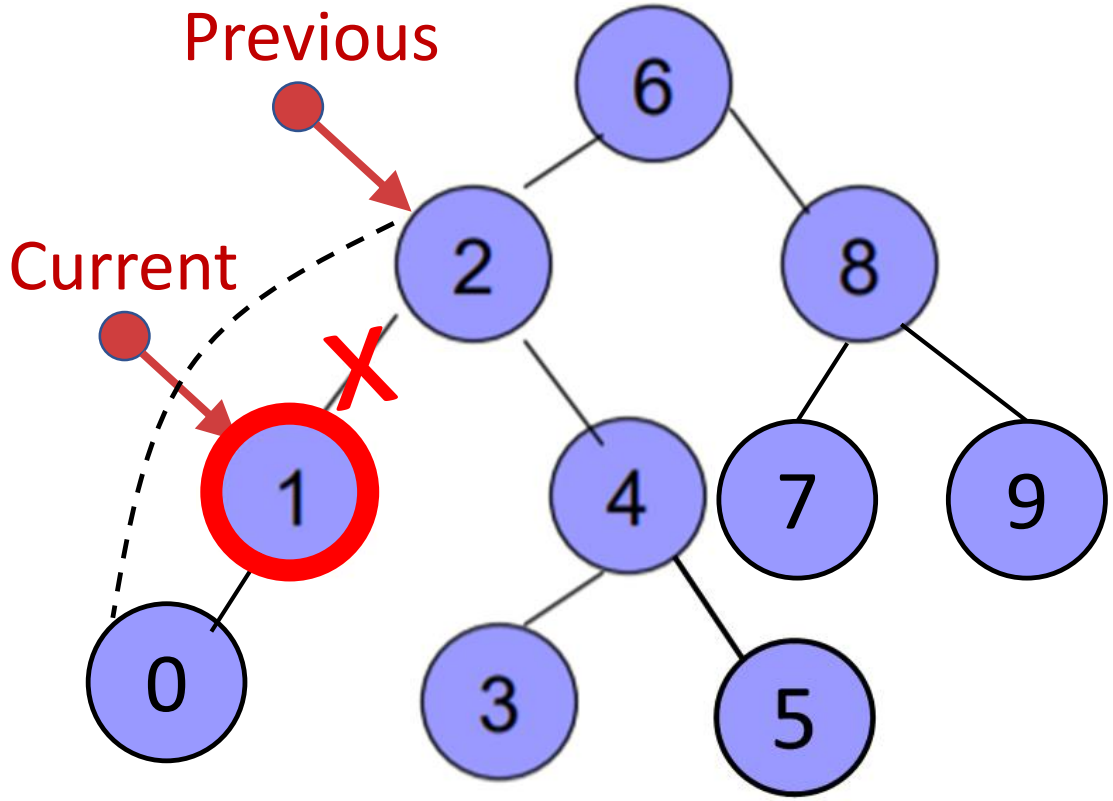
BST – Deletion

Case 2: One child

Delete ‘1’

```
if (Previous->right == Current)
        If(Current->right!=NULL)
                Previous->right= Current->right;
        else
                Previous->right= Current->left;

else
        If(Current->right!=NULL)
                Previous->left= Current->right;
        else
                Previous->left= Current->left;
```

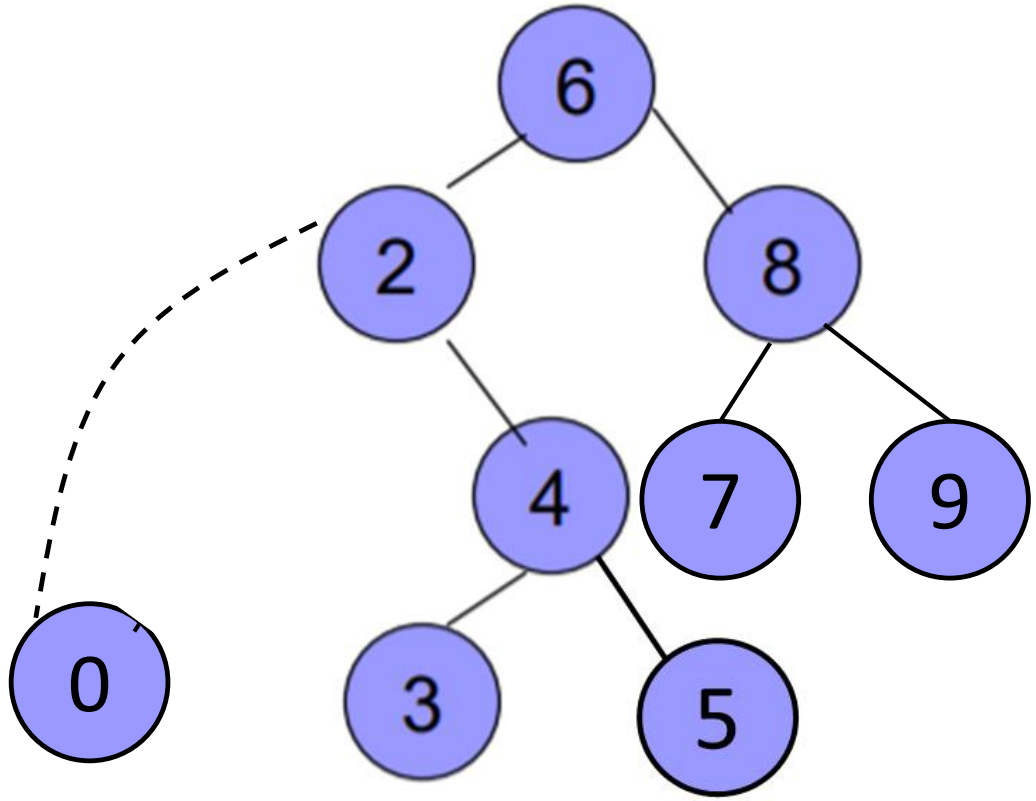


BST – Deletion

Case 2: One child

Delete ‘1’

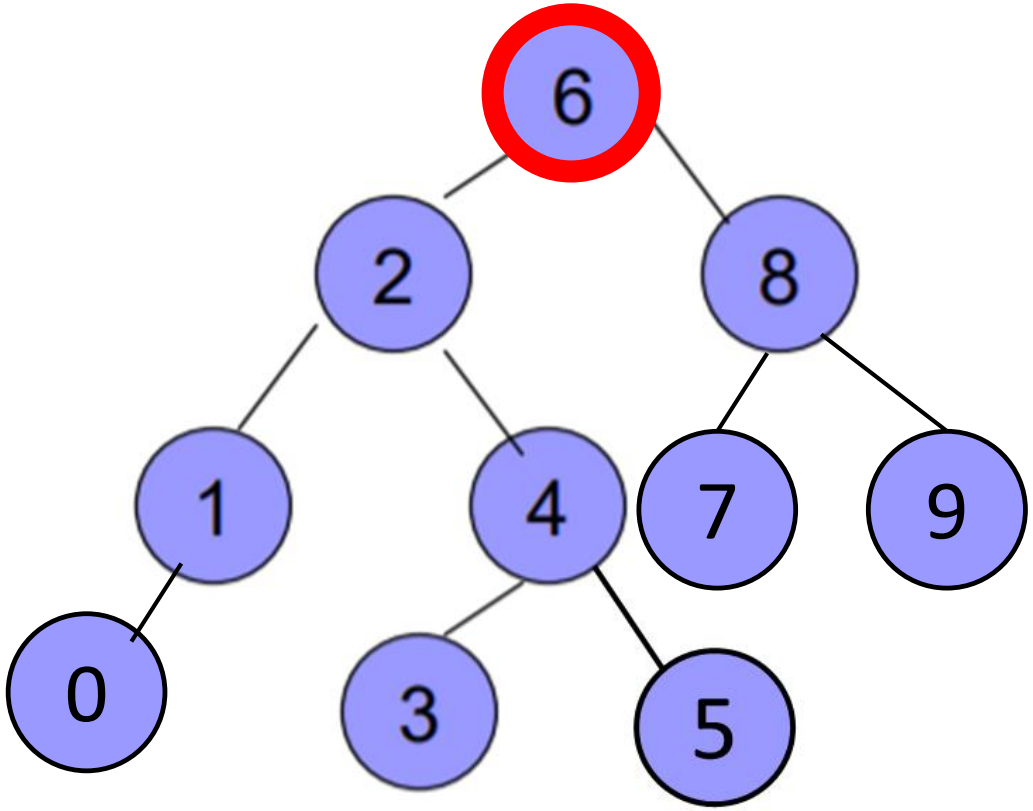
Delete Current



BST – Deletion

Case 3: Two Children

Delete ‘6’

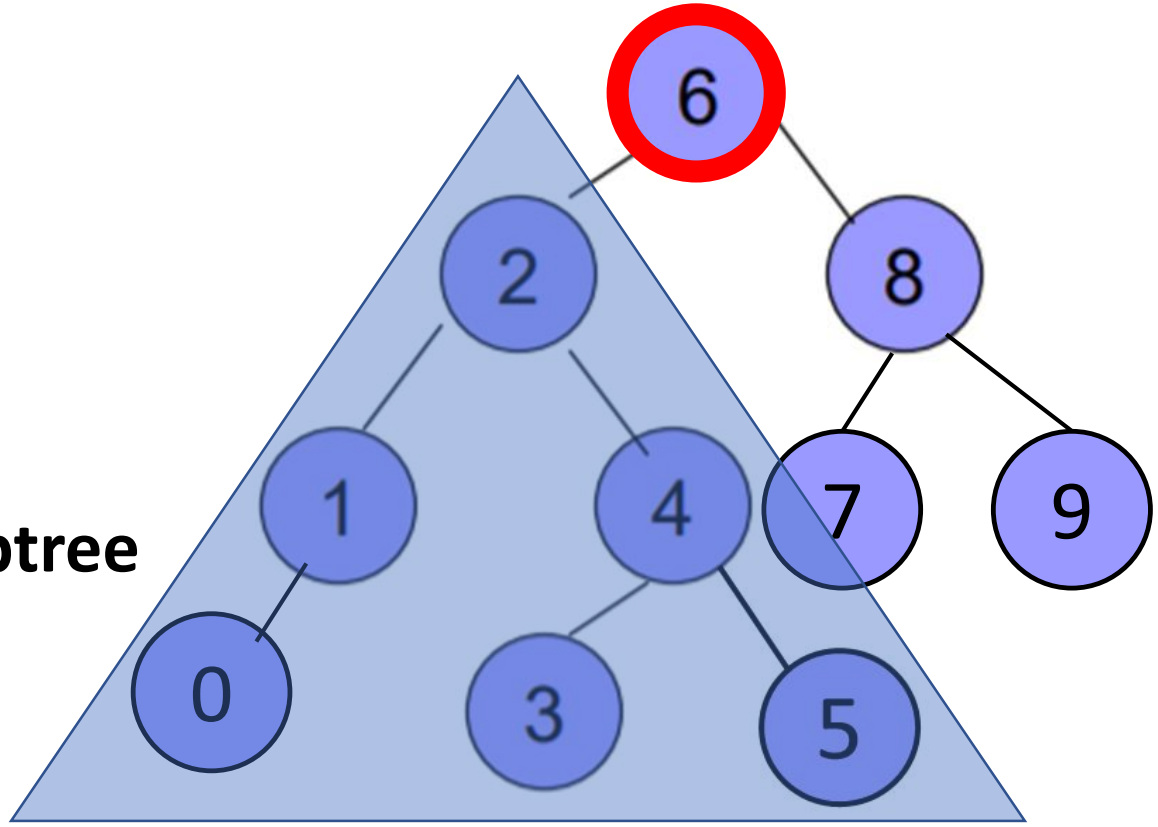


BST – Deletion

Case 3: Two Children

Delete ‘6’

Find maximum in left subtree

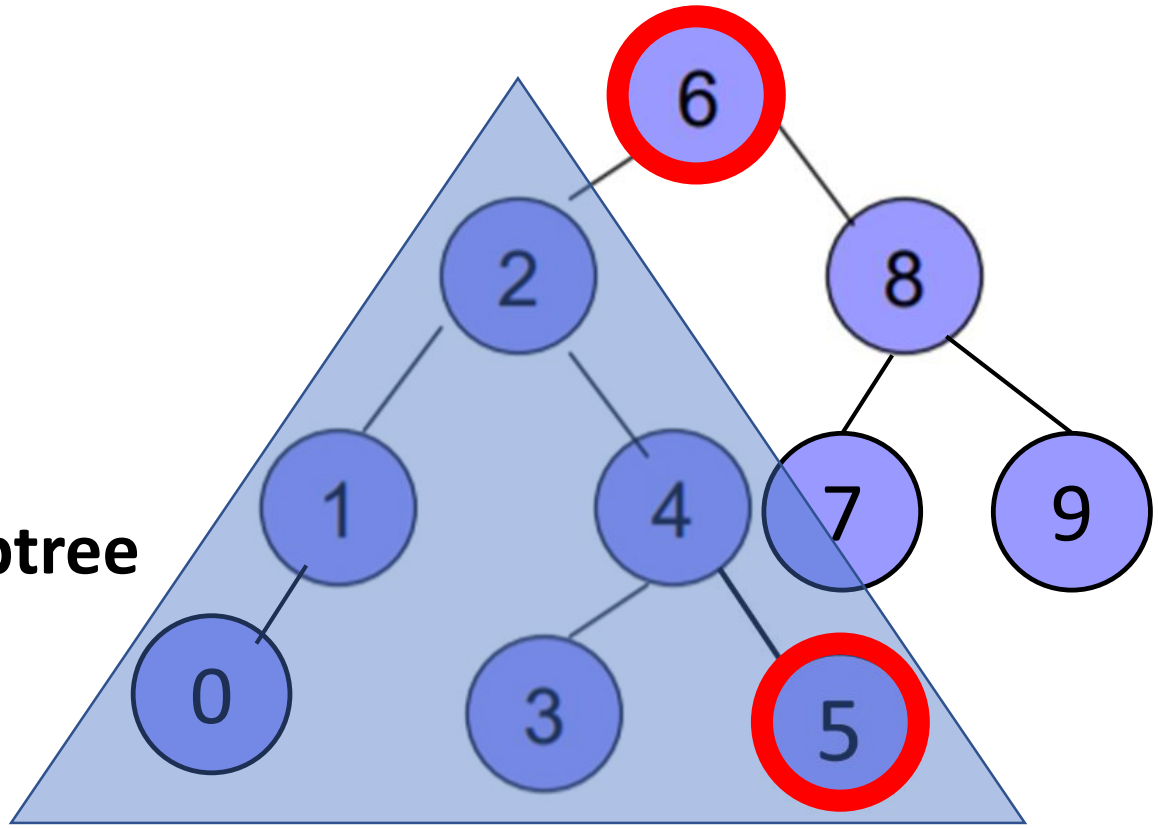


BST – Deletion

Case 3: Two Children

Delete ‘6’

Find maximum in left subtree

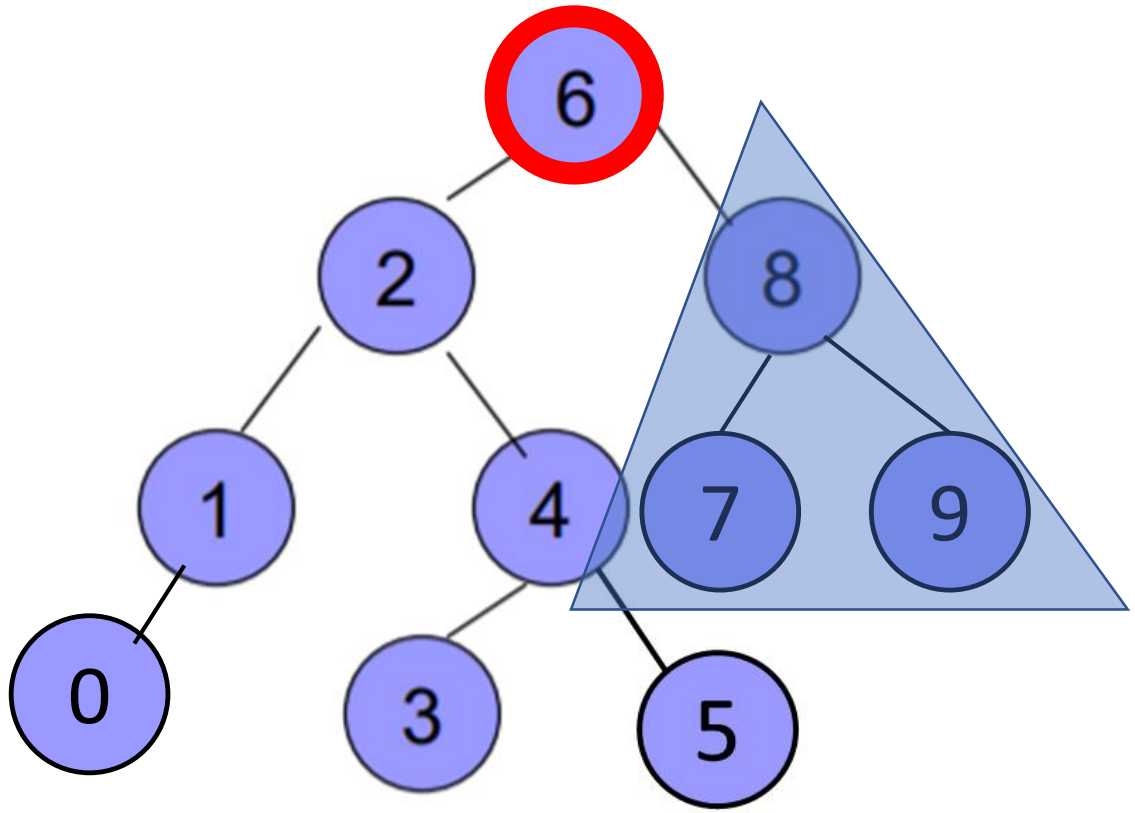


BST – Deletion

Case 3: Two Children

Delete ‘6’

Find minimum in left subtree

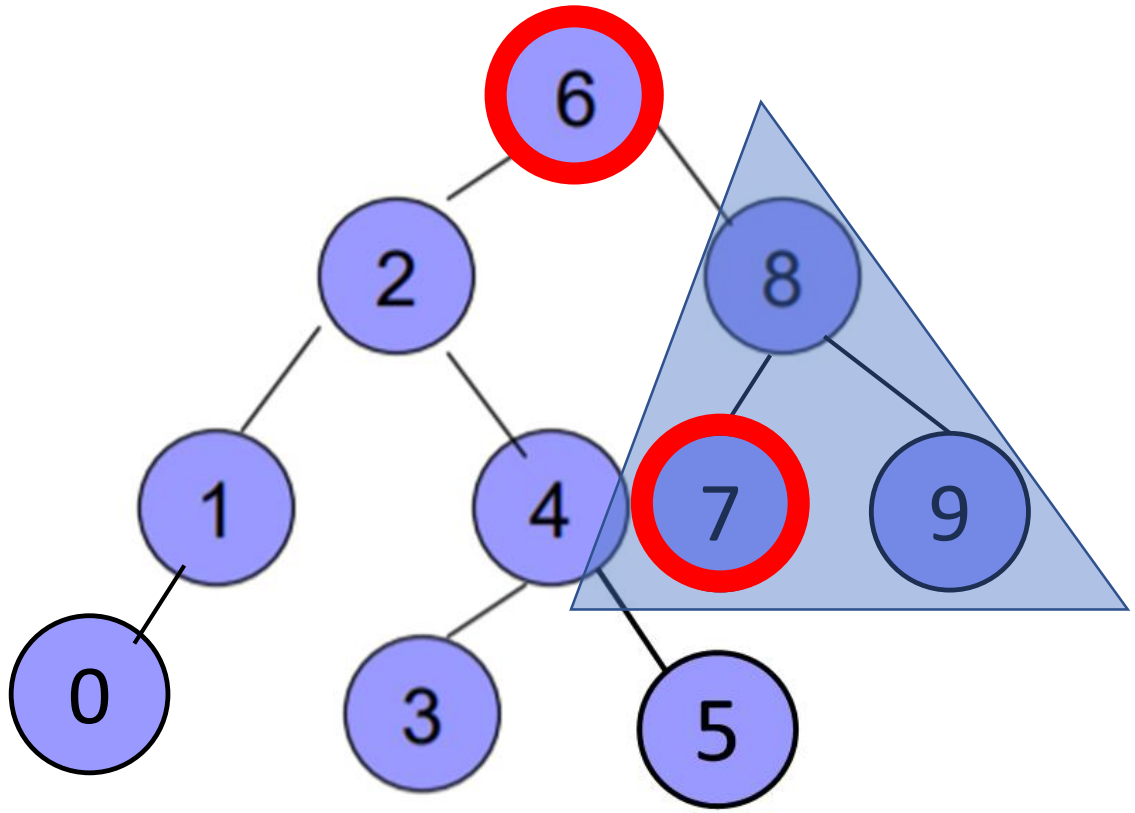


BST – Deletion

Case 3: Two Children

Delete ‘6’

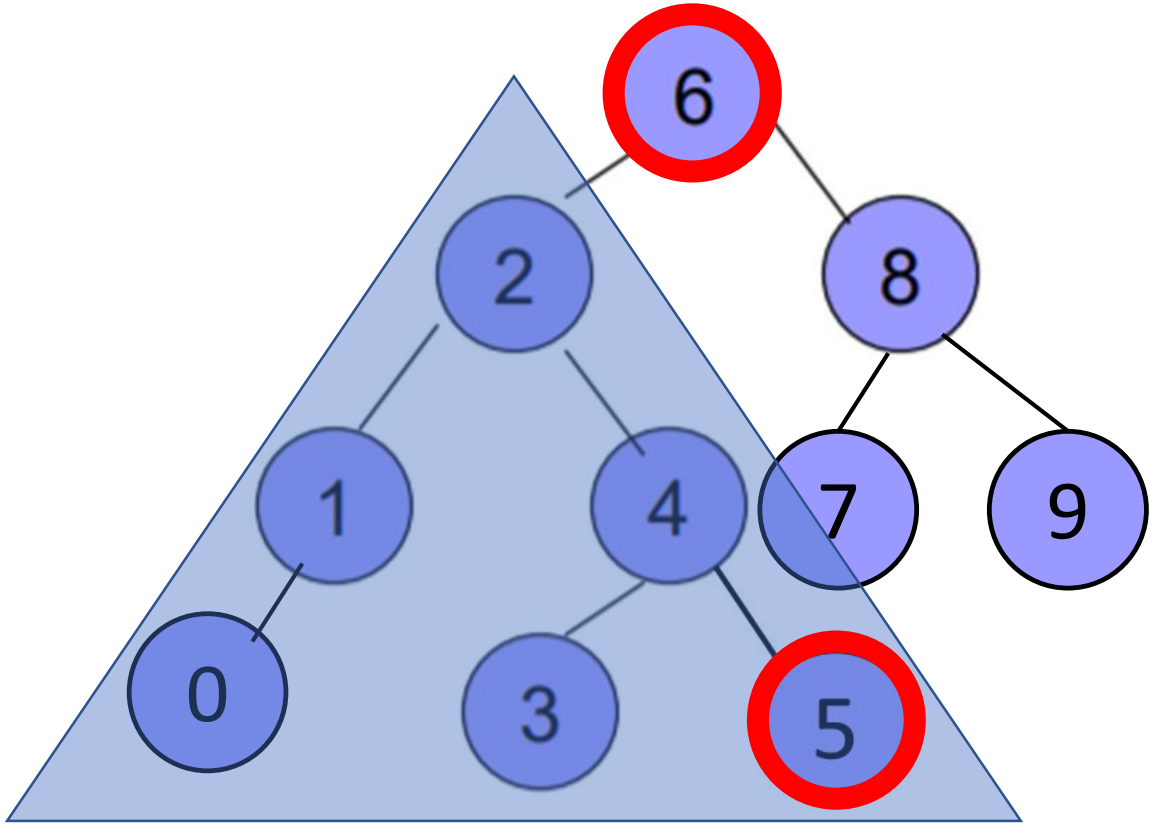
Find minimum in left subtree



BST – Deletion

Case 3: Two Children

Delete ‘6’

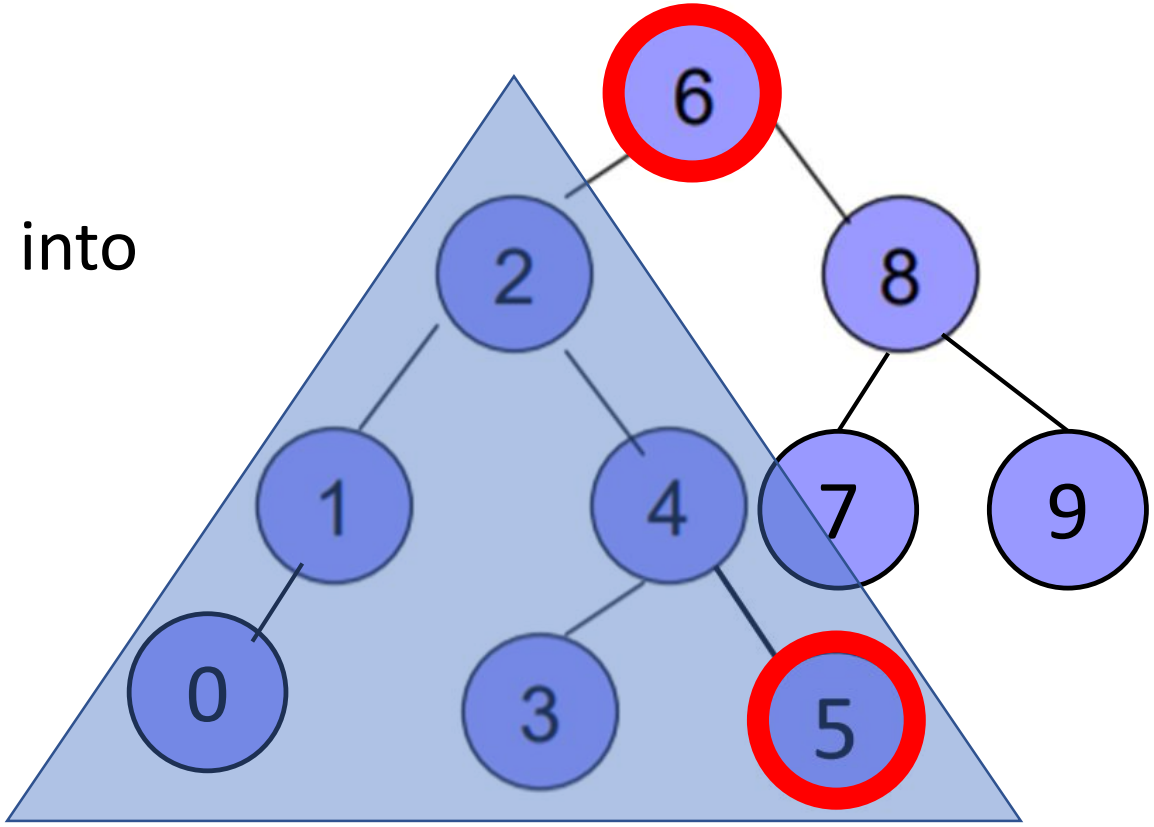


BST – Deletion

Case 3: Two Children

Delete ‘6’

- Simply copy the contents of ‘**5**’ into ‘**6**’ including key

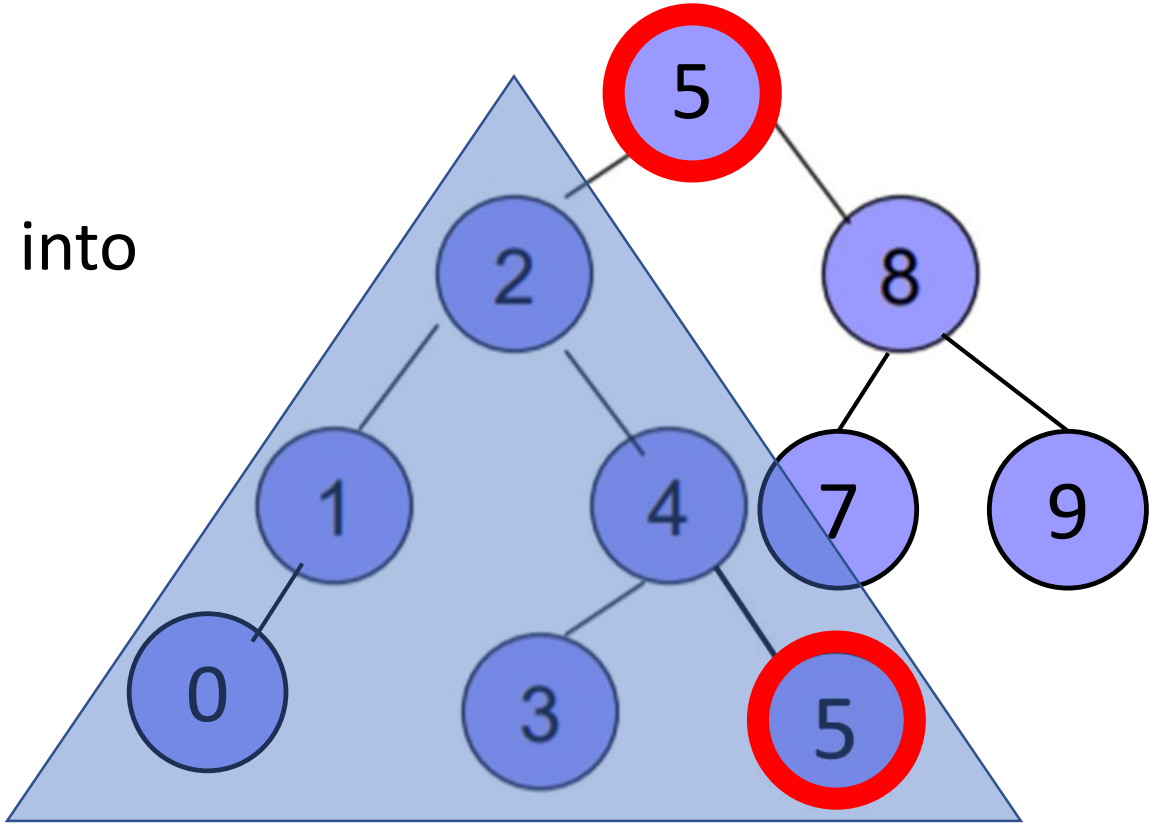


BST – Deletion

Case 3: Two Children

Delete ‘6’

- Simply copy the contents of ‘**5**’ into ‘**6**’ including key

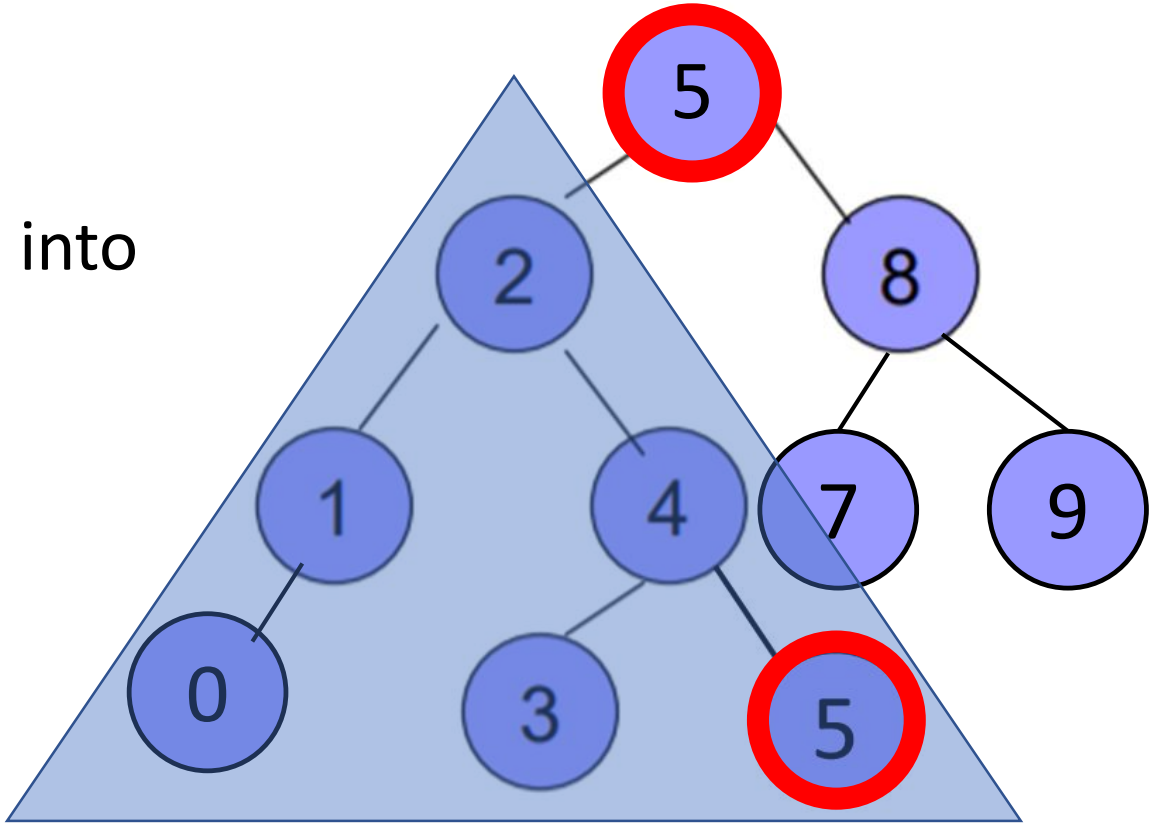


BST – Deletion

Case 3: Two Children

Delete ‘6’

- Simply copy the contents of ‘**5**’ into ‘**6**’ including key
- Delete ‘5’ in left subtree

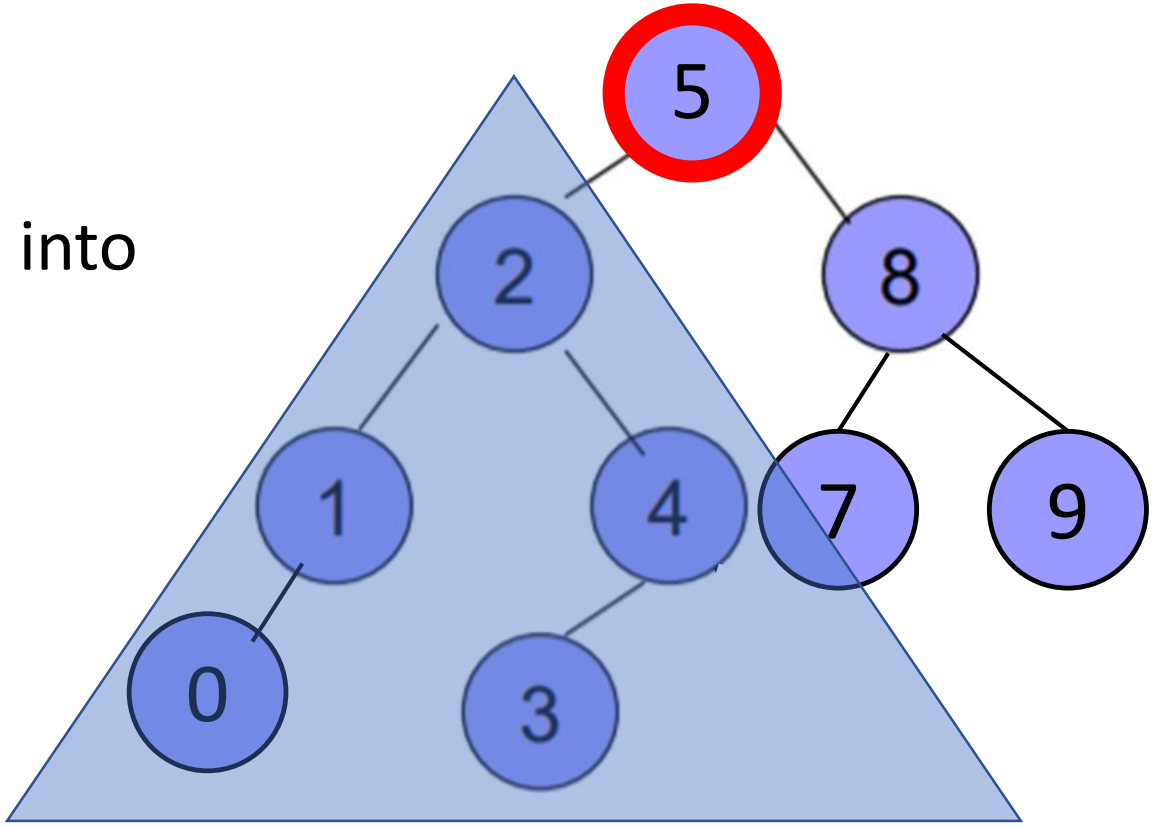


BST – Deletion

Case 3: Two Children

Delete ‘6’

- Simply copy the contents of ‘5’ into ‘6’ including key
- Delete ‘5’ in left subtree

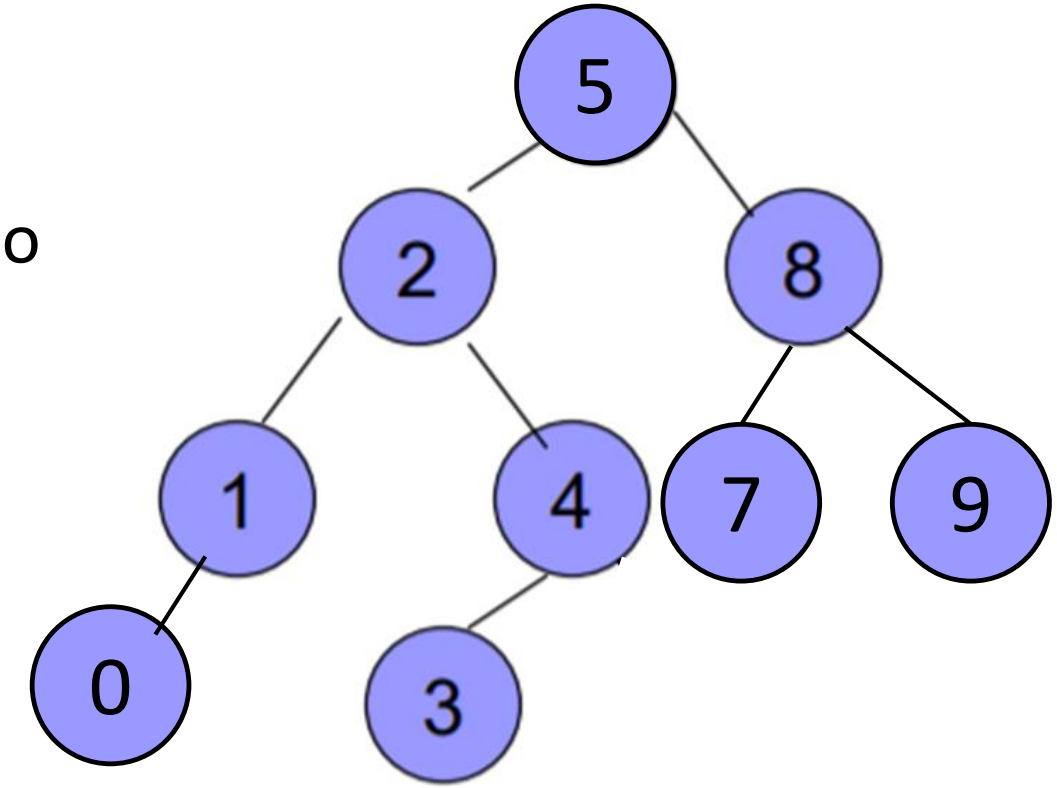


BST – Deletion

Case 3: Two Children

Delete ‘6’

- Simply copy the contents of ‘5’ into ‘6’ including key
- Delete ‘5’ in left subtree



Thanks a lot

If you are taking a Nap, **wake up**.....Lecture Over