Lab 11: Binary Search Tree (BST) Deletion

CLO:

02

Objectives:

In this lab you will learn about Deletion in BST

Removing a node in Binary Search Tree

Remove operation on binary search tree is more complicated, than add and search. Basically, in can be divided into two stages:

1. **Search for a node to remove**

   If the node is found, run remove algorithm.

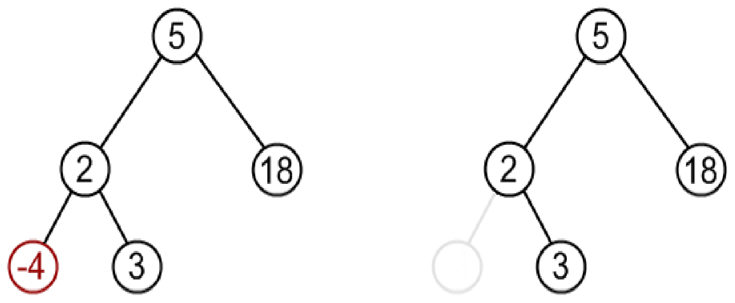
2. **Remove algorithm in detail**

   Now, let's see more detailed description of a remove algorithm. First stage is identicalto algorithm for lookup, except we should track the parent of the current node. Second part is trickier. There are three cases, which are described below.

I. **Node to be removed has no children.**

This case is quite simple. Algorithm sets corresponding link of the parent to NULL and disposes the node.

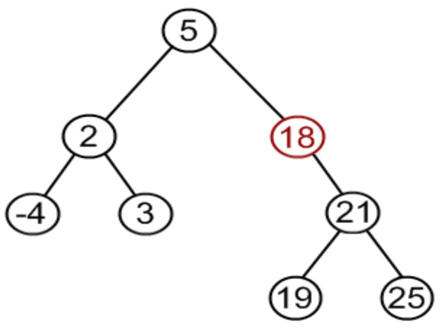
Example, Remove -4 from a BST.

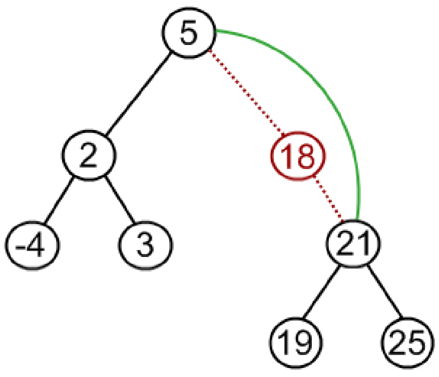


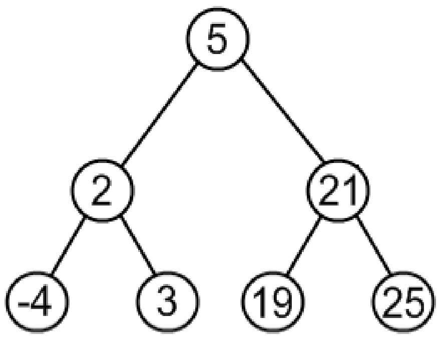
II. **Node to be removed has one child.**

It this case, node is cut from the tree and algorithm links single child (with it's sub tree) directly to the parent of the removed node.

Example. Remove 18 from a BST.

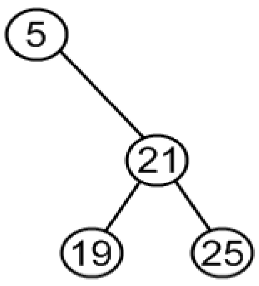


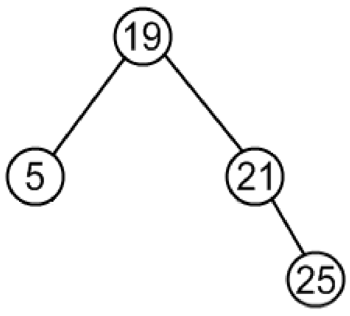




III. **Node to be removed has two children.**

This is the most complex case. To solve it, let us see one useful BST property first. We are going to use the idea, that the same set of values may be represented as different binary-search trees.





For **example,** those BSTs: contains the same values {5, 19, 21, 25}. To transform first tree into second one, we can do following:

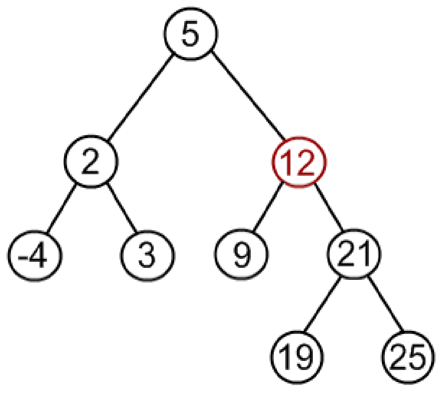
Choose minimum element from the right sub tree (19 in the example);

- Replace 5 by 19;
- Hang 5 as a left child.

The same approach can be utilized to remove a node, which has two children:

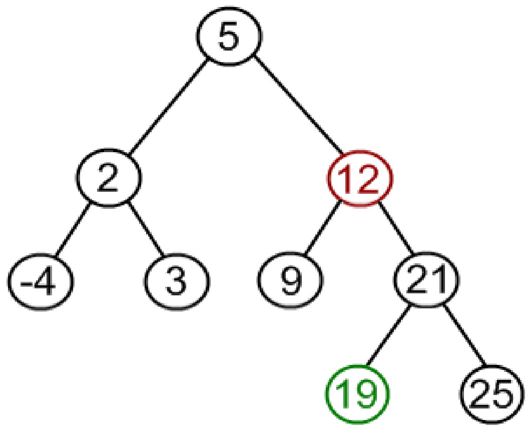
- Find a minimum value in the right sub tree;
- Replace value of the node to be removed with found minimum. Now, right sub tree contains a duplicate!
- Apply remove to the right sub tree to remove a duplicate.
- Notice, that the node with minimum value has no left child and, therefore, its removal may result in first or second cases only.

Example. Remove 12 from a BST.

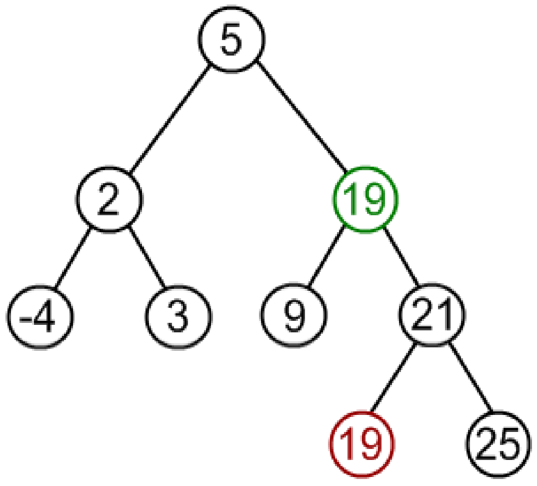


Steps:

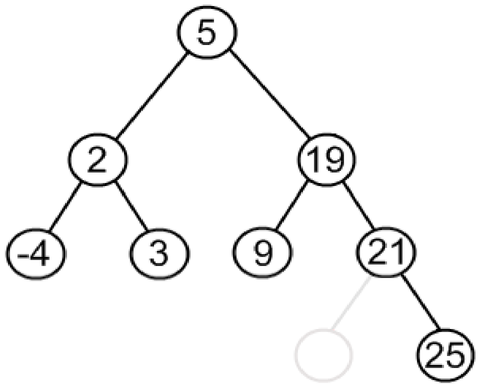
1. Find minimum element in the right sub tree of the node to be removed. In current example it is 19.



2. Replace 12 with 19. Notice, that only values are replaced, not nodes. Now we have two nodes with the same value.



3. Remove 19 from the left sub tree.



Lab Tasks

1. You are tasked with implementing a Patient Management System for a hospital using a Binary Search Tree (BST). The system should efficiently manage patient records based on unique Patient IDs.

Task:

- BST Insertion: Implement a function to insert patient records into the BST based on Patient IDs. Write functions to find the min and max age value in BST.
- BST Deletion: Write a function to implement all the three cases of deletion in BST.
- BST Sorting: As we know in order traversal prints the data in sorted order, but the data is printed in ascending order. Your task is to create a new traversal which will print the data in descending order.
- BST Height: Implement a function to find the height of a binary search tree.
- BST Display: Write a function that display the tree data in level order.

2. Consider the following Binary Search Tree (BST):

```
         50
        /    \
      30   70
     /  \     /  \
   20  40 60  80
    / \
 10  25
```

Perform the following deletion operations on the given BST and show the resulting tree after each deletion:
Delete the node with key 30.
Delete the node with key 70.
Delete the node with key 50.
For each deletion, provide the resulting tree structure after the operation.