CSC 148: Introduction to Computer Science

Week 9

BSTs mutation

Insert / Delete operations

University of Toronto Mississauga,

Department of Mathematical and Computational Sciences

Delete

- Hint: pull out the tree deletion worksheet from last week ...

- Input:
  - An item that we wish to delete from the BST (if it exists!)

- Outcomes:
  - if item found => remove the first occurrence of this item from the BST
    - This node gets “disconnected” / ”extracted” / “removed” from the BST
  - if item not found => do nothing

Delete

```
def delete(self, item: Any) -> None:
    """Remove *one* occurrence of <item> from
    this BST.

    Do nothing if <item> is not in the BST.
    """
```

Deletion of an Item from BST

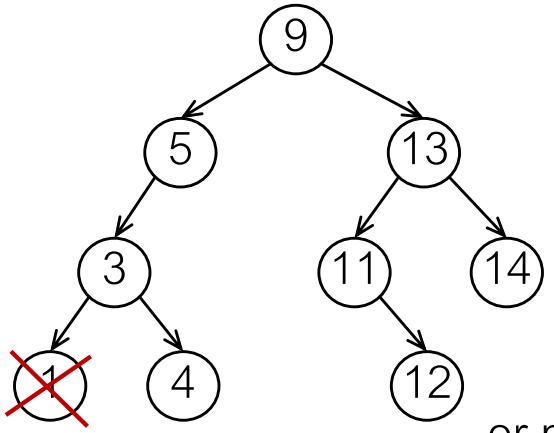
- Locate the item to delete, by traversing the tree
- Say that self is the current subtree being inspected
  - What to do if the BST is empty (self._root is None)?
  - What if item to delete is less than the value self._root?
  - What if item to delete is more than the value self._root?
  - What if item to delete equals the value self._root?

Deletion of an Item from a BST

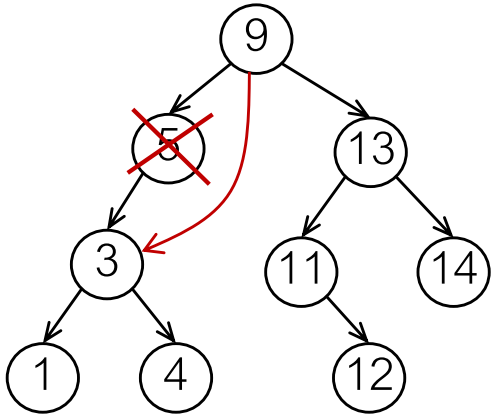
- Worksheet....
  - Go through the questions step-by-step
  - Discuss your answers with your neighbours
  - As usual, we will discuss this together once you're done

Deletion of an Item from a BST

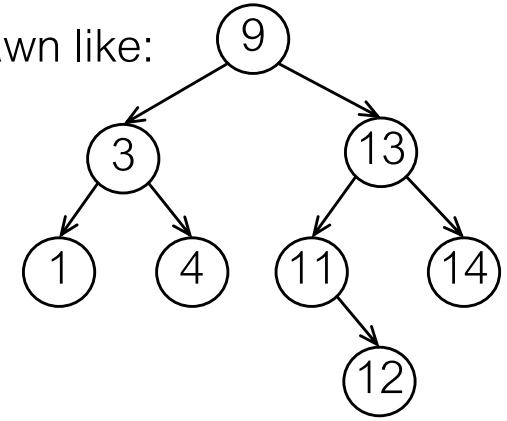
- Item found => time to remove the node
- Three cases – first two are pretty straightforward though:
  - 1. It's a leaf (Has no children)
    - e.g., node with value 1



  - 2. Has only one child
    - e.g., node with value 5
    - Does it matter which child?

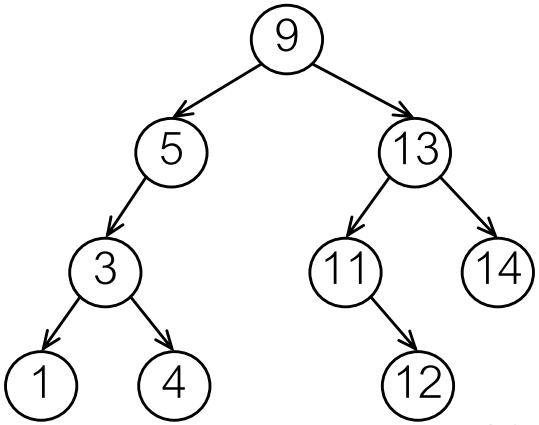


or more nicely drawn like:

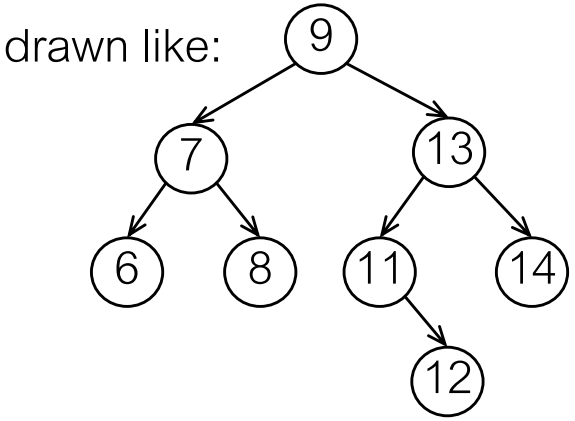


Deletion of an Item from a BST

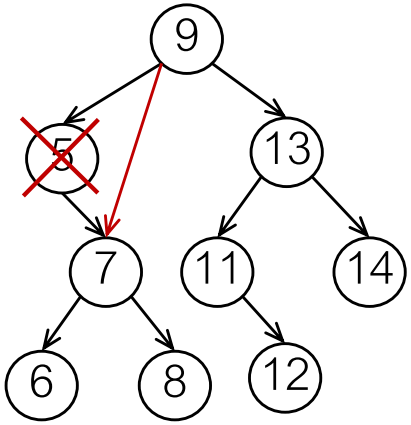
- Item found => time to remove the node
- Three cases – first two are pretty straightforward though:
  - 1. It's a leaf (Has no children)
    - e.g., node with value 1



or more nicely drawn like:



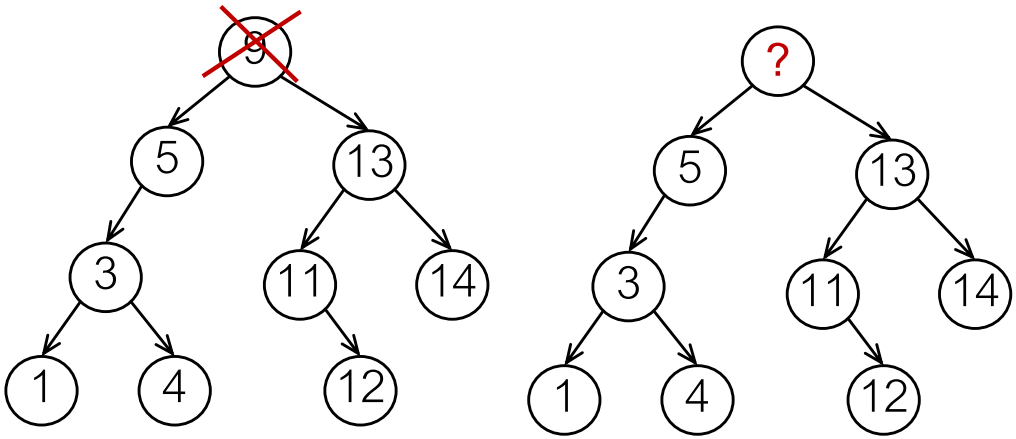
  - 2. Has only one child
    - e.g., node with value 5



    - Does it matter which child?

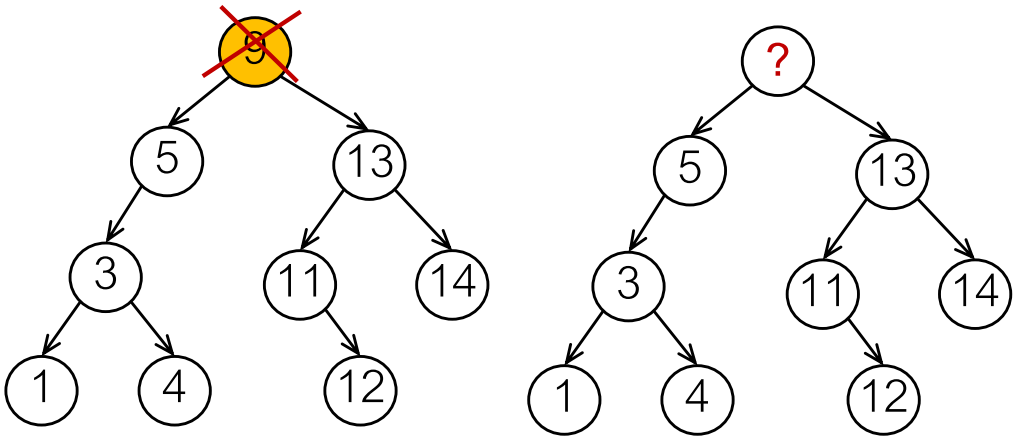
Deletion of Data from BST?

- 3. Has both children => a bit more complex
  - BST properties must still hold after the deletion! For each node X:
    - P1) every node in the left subtree must have a value smaller than X's value
    - P2) every node in the right subtree must have a value larger than X's value
  - Let's say we want to remove node with value 9
    - Idea: Keep the node, clear value 9, pick another value from under this node (as a replacement), and remove THAT node.
    - How do we do this, while still keeping the BST properties? Thoughts?



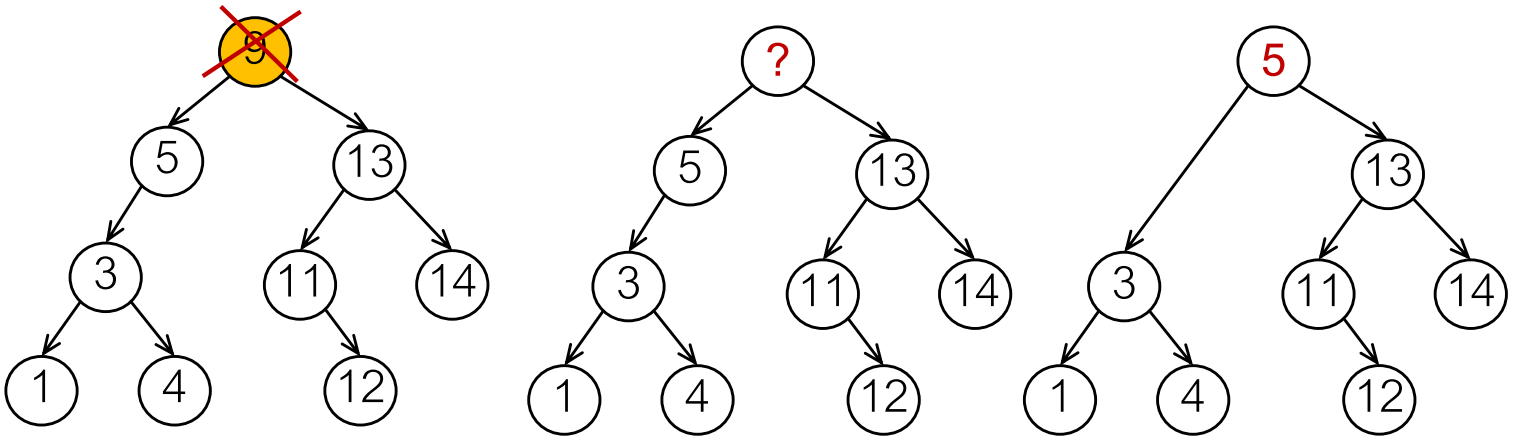
Deletion of Data from BST?

- 3. Has both children => a bit more complex
  - BST properties must still hold after the deletion! For each node X:
    - P1) every node in the left subtree must have a value smaller than X's value
    - P2) every node in the right subtree must have a value larger than X's value
  - Let's say we want to remove node with value 9
    - Idea: Keep the node, clear value 9, pick another value from under this node (as a replacement), and remove THAT node.
    - What if we picked a value from the LEFT subtree? Which one makes sense?



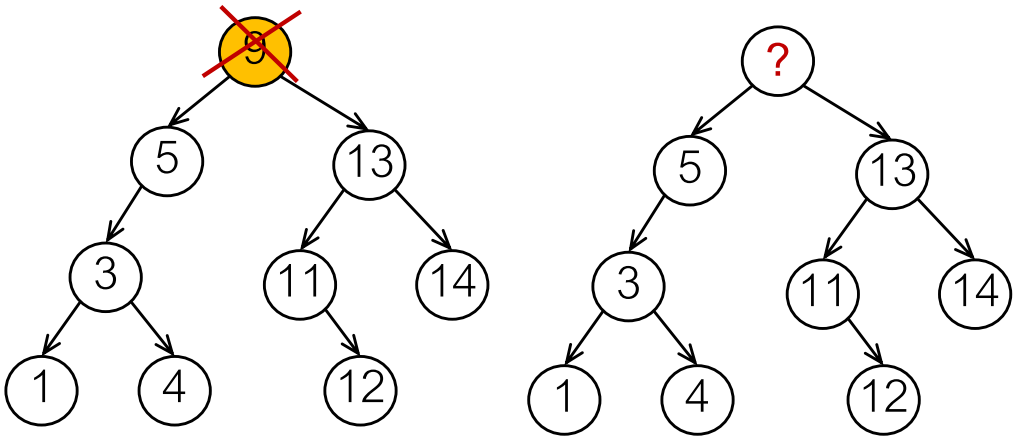
Deletion of Data from BST?

- 3. Has both children => a bit more complex
  - BST properties must still hold after the deletion! For each node X:
    - P1) every node in the left subtree must have a value smaller than X's value
    - P2) every node in the right subtree must have a value larger than X's value
  - Let's say we want to remove node with value 9
    - Idea: Keep the node, clear value 9, pick another value from under this node (as a replacement), and remove THAT node.
    - What if we picked a value from the LEFT subtree? Which one makes sense?



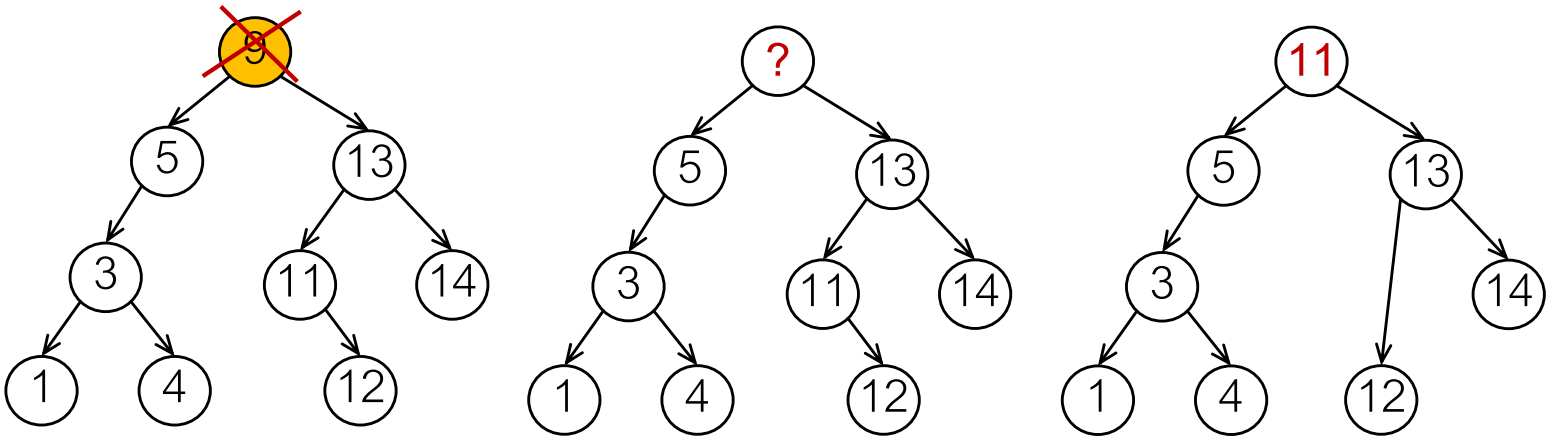
Deletion of Data from BST?

- 3. Has both children => a bit more complex
  - BST properties must still hold after the deletion! For each node X:
    - P1) every node in the left subtree must have a value smaller than X's value
    - P2) every node in the right subtree must have a value larger than X's value
  - Let's say we want to remove node with value 9
    - Idea: Keep the node, clear value 9, pick another value from under this node (as a replacement), and remove THAT node.
    - What if we picked a value from the RIGHT subtree? Which one makes sense?



Deletion of Data from BST?

- 3. Has both children => a bit more complex
  - BST properties must still hold after the deletion! For each node X:
    - P1) every node in the left subtree must have a value smaller than X's value
    - P2) every node in the right subtree must have a value larger than X's value
  - Let's say we want to remove node with value 9
    - Idea: Keep the node, clear value 9, pick another value from under this node (as a replacement), and remove THAT node.
    - What if we picked a value from the RIGHT subtree? Which one makes sense?



Delete - Recap

- Traverse the tree to locate the node with the intended value
- If we reach a leaf and no match => not found, done!
- If value to delete is smaller => inspect left subtree
- If value to delete is larger => inspect right subtree
- If value to delete found (equal)
  - Case a) No children => easy, just remove the node
  - Case b) One child => easy, just connect the child to current subtree's parent
  - Case c) Two children => clear the value, pick a replacement value from a descendant under it, and remove that descendant node
    - Max from left subtree, or min from right subtree

Implementation ...

- Onto Pycharm ...

Insert

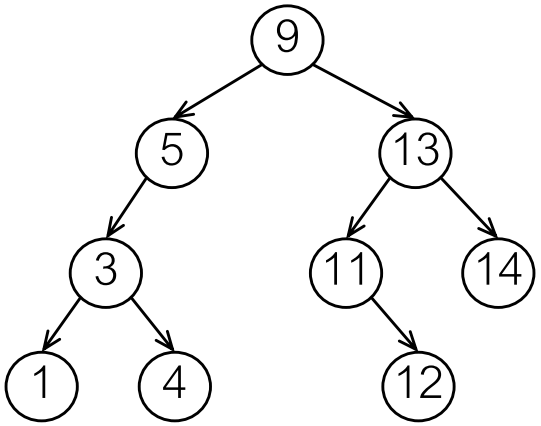
- Insert must ensure BST condition holds:
  - Left subtree of N => smaller values than N's data
  - Right subtree of N => larger values than N's data

- How do we insert a new node then?

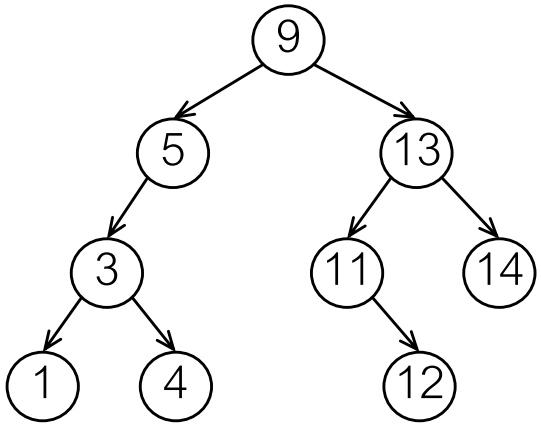
Insert

- Where do we insert a new value, while keeping it a BST? Thoughts?

insert value 7?



insert value 10?



Implementation ...

- You will implement this in this week's lab...