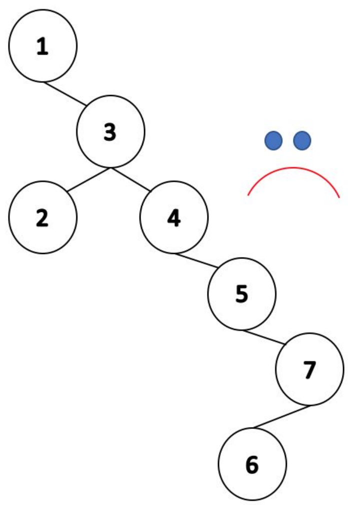
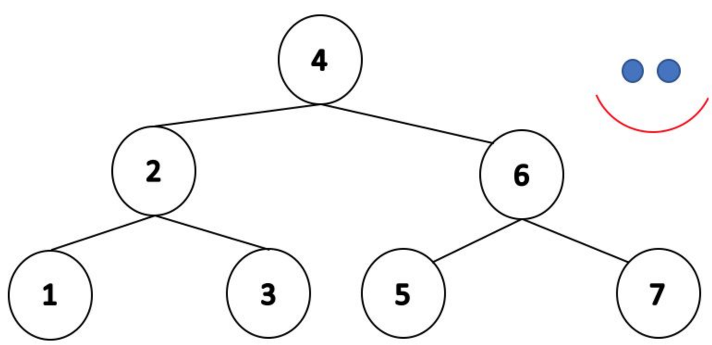
CS 225 Spring 2019 :: TA Lecture Notes
2/22 BST Remove

By Wenjie

- **Height of a BST**
    - between O(lg n) and O(n). O(lg n) is what we want!
    - The height depends on the insert order:
        - ex: 1 3 2 4 5 7 6
          Slow: _find takes O(n)



        - ex: 4 2 3 6 7 1 5
          Fast: _find takes O(log n)



    - There are **n!** orders to input **n** elements
    - What is the average height among all trees with different input order?
        - **h** is about **log n**
        - Intuition:

CS 225 Spring 2019 :: TA Lecture Notes
2/22 BST Remove

By Wenjie

        - Exactly two input order yields the worst case: linked-list-like trees
          1 2 3 4 5 6 7   and   7 6 5 4 3 2 1
        - If 4 is the root, the tree is relatively balanced. There are 6! input orders that yield 4 as the root.
        - Therefore, averagely the tree should be relatively balanced.
    - **BUT**, when the data is ordered (which is usually good), we get our worst case!

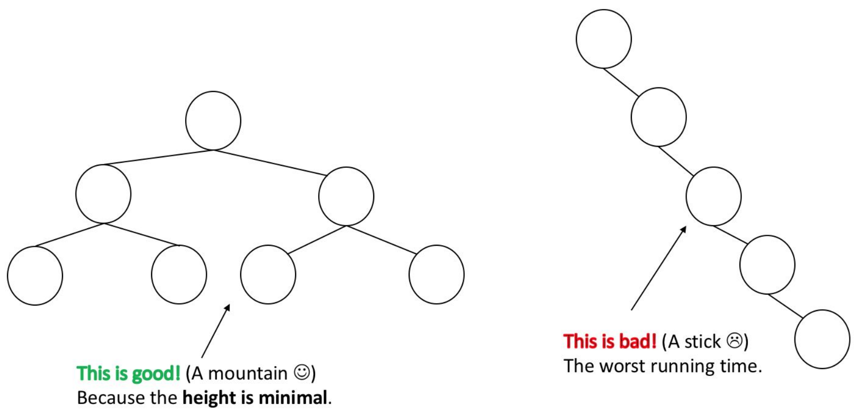
Operation	BST Average case	BST Worst case	Sorted Array	Sorted List
find	O(lg n)	O(h) <= O(n)	O(lg n) with binary search	O(n) no binary search
insert	O(lg n)	O(h) <= O(n)	O(n) find data with O(lg n), move the data O(n)	O(n) find data with O(n)
delete	O(lg n)	O(h) <= O(n)	O(n)	O(n)
traverse	O(n)	O(n)	O(n)	O(n)

    - BST out-performs array/linked-list, but the worst cases are worrying (especially with sorted data);
        - We need to make sure that we never have the worst case on BST!

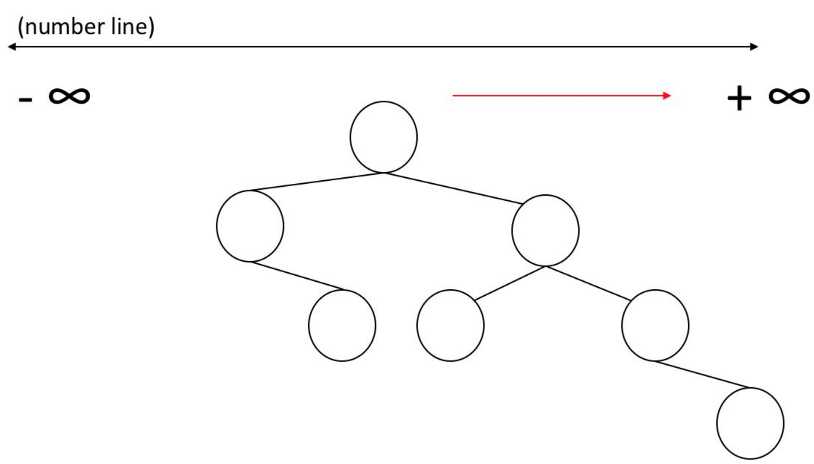
- **Height-Balanced Tree**
    - We want mountains over sticks

CS 225 Spring 2019 :: TA Lecture Notes
2/22 BST Remove

By Wenjie



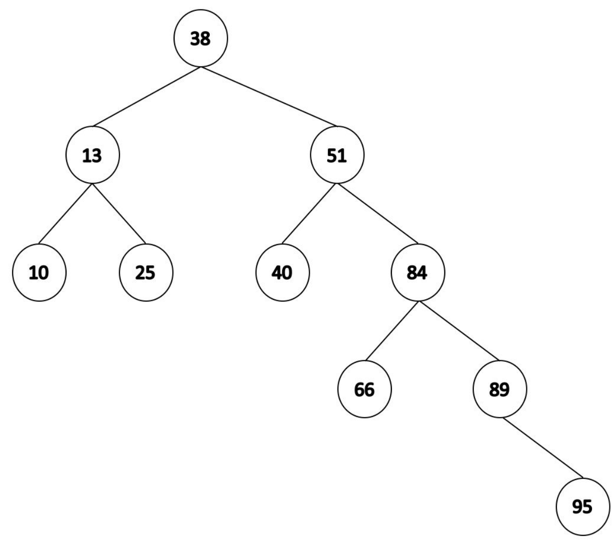
- Balance Factor: b = height(T_R) - height(T_L)
  - Left heavy trees: balance factor negative
  - Right heavy trees: balance factor positive



- A tree is height balanced if **|b| <= 1**
- Balance is determined locally
  b(95) = 0 → no children; b(89) = 1; b(84) = 2 - 1 = 1; b(40) = 0; b(51) = 3 - 1 = 2; b(38) = 4 - 2 = 0

CS 225 Spring 2019 :: TA Lecture Notes
2/22 BST Remove

By Wenjie



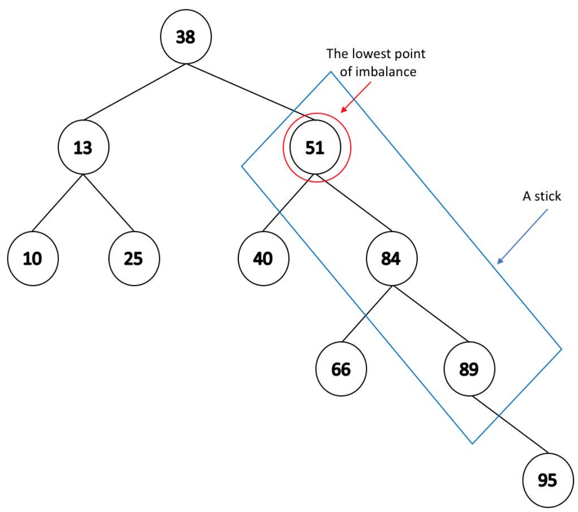
- **The lowest point of imbalance** is the node 51. It is the deepest node in the tree that is out of balance. The tree starting at 51 looks like a stick.
    - We want to turn sticks into mountains → break into half and join (almost like folding) while preserving the BST structure.

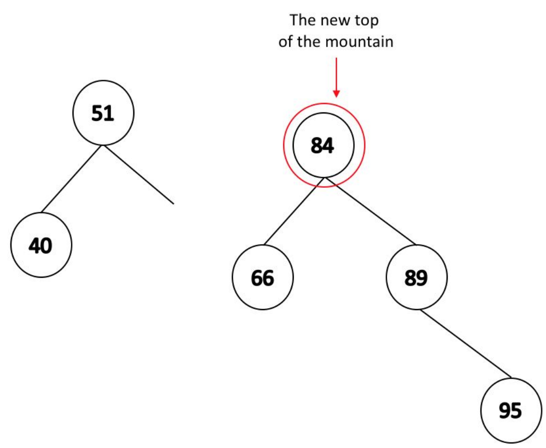
- **BST rotation**
    - A method to solve the imbalance:
        1. Must maintain BST properties
        2. Must be locally performable in O(1) time

1. Find the imbalance and the stick;   2. Break and fold the stick

CS 225 Spring 2019 :: TA Lecture Notes
2/22 BST Remove

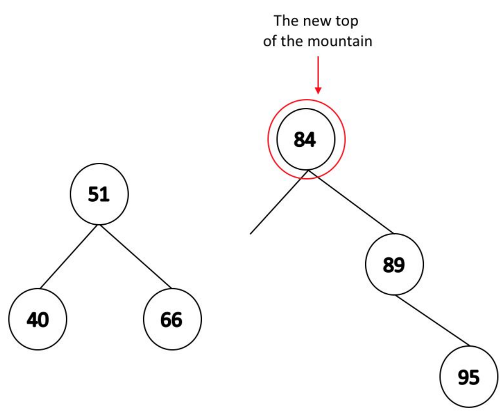
By Wenjie

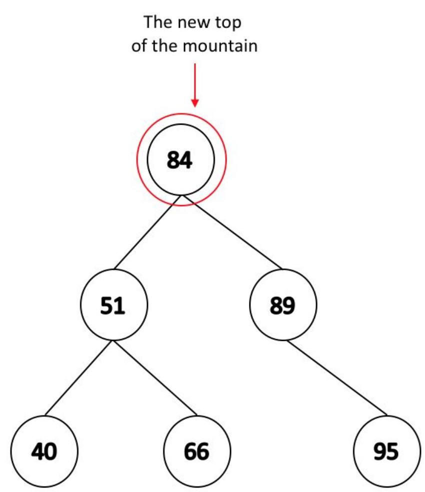




3. Move the lost element (66)

4. Create the mountain





- A picture comparing the tree before the rotation and after the rotation

CS 225 Spring 2019 :: TA Lecture Notes
2/22 BST Remove

By Wenjie