Useful Algorithms

# Boyer-Moore Majority Voting Algorithm

* Find majority occurring element in a list
* O(n) time (2 list traversals), O(1) memory
* Array Algorithm

The **Boyer-Moore voting** algorithm is one of the popular optimal algorithms which is used to find the majority element among the given elements that have more than N/ 2 occurrences. This works perfectly fine for finding the majority element which takes **2 traversals over the given elements**, which works in O(N) time complexity and O(1) space complexity.

**Input :**{1,1,1,1,2,3,5}

**Output :** 1

**Explanation :** 1 occurs more than 3 times.

**Input :** {1,2,3}

**Output :** -1

**Steps to implement the algorithm :**  
**Step 1 (first for loop) –** Find a candidate with the majority –

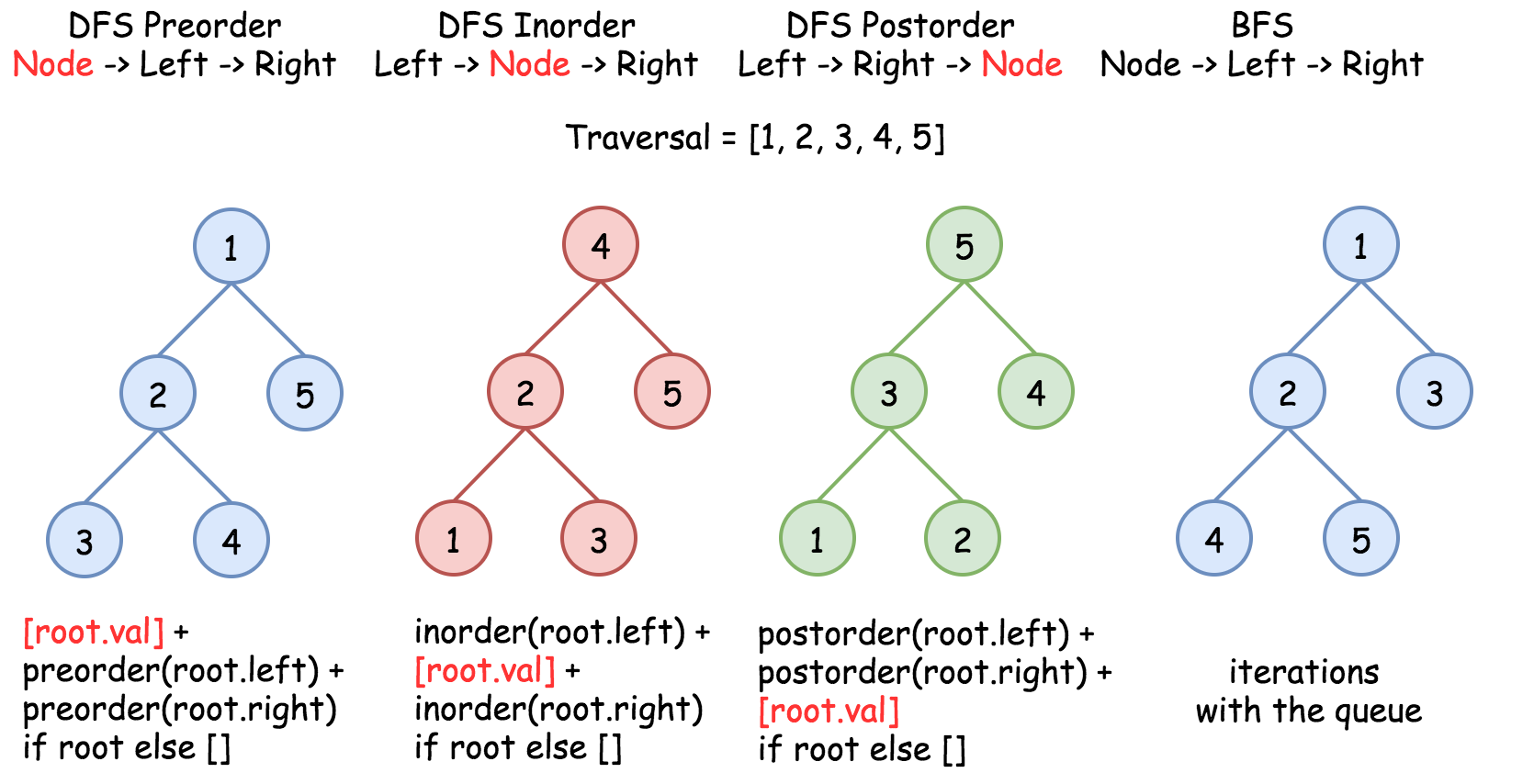
* Initialize a variable say **i ,votes = 0, candidate =-1**
* Traverse through the array using for loop
* If **votes = 0,** choose the **candidate = arr[i]** , make **votes=1**.
* else if the current element is the same as the candidate increment votes
* else decrement votes.

**Step 2 (the second for loop) –** Check if the candidate has more than N/2 votes –

* Initialize a variable count =0 and increment count if it is the same as the candidate.
* If the count is >N/2, return the candidate.
* else return -1.

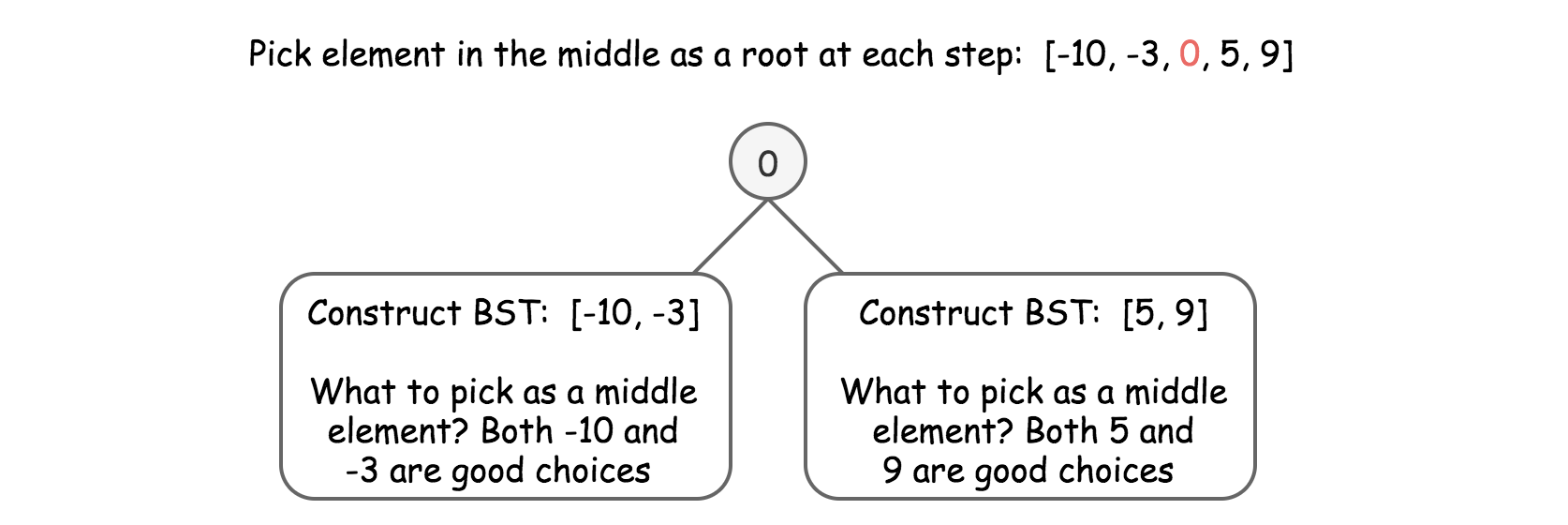
# Build Balanced BST from Sorted Array

Can choose either dfs of bfs for traversal. Will use dfs preorder as its easiest to understand.



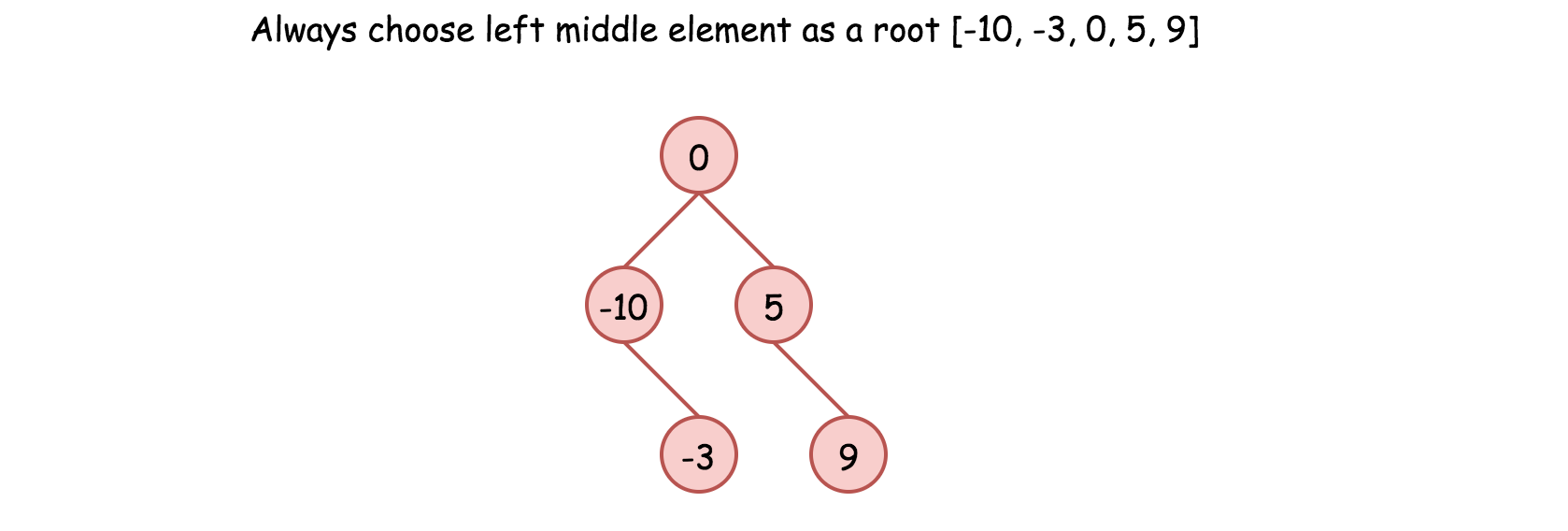
Since we want a balanced tree, with every recursive iteration, pick the middle node in the array passed.

Basically, the height-balanced restriction means that at each step one has to pick up the number in the middle as a root. That works fine with arrays containing an odd number of elements but there is no predefined choice for arrays with an even number of elements.



#### Approach 1: Preorder Traversal: Always Choose Left Middle Node as a Root

**Algorithm**



* Implement helper function helper(left, right), which constructs BST from nums elements between indexes left and right:
  + If left > right, then there is no elements available for that subtree. Return None.
  + Pick left middle element: p = (left + right) // 2.
  + Initiate the root: root = TreeNode(nums[p]).
  + Compute recursively left and right subtrees: root.left = helper(left, p - 1), root.right = helper(p + 1, right).
* Return helper(0, len(nums) - 1).

**Complexity Analysis**

* Time complexity: O(N) since we visit each node exactly once.
* Space complexity: O(logN)

The recursion stack requires O(logN) space because the tree is height-balanced. Note that the O(N) space used to store the output does not count as auxiliary space, so it is not included in the space complexity.

# Floyds Cycle Algorithm

An algorithm that allows you to detect if there is a cycle in a linked list without requiring the use of additional memory. See leetcode 141.

# Sliding Window

Array/string algorithm.

A window is a range of elements an array/string which usually defined by the start end indices [I,j) (left closed, right open).

# Python Tricks

## Iterate through individual digits of a number

Ver1:

A screenshot of a computer code

Description automatically generated

Ver2:

A computer screen shot of a code

Description automatically generated

The digit variable, you can grab an individual digit you are looking. In ver 1, the digit you grab from the number sumval is 0 indexed by index starting from the right most digit.

Ex: 54321 // 10\*\***6** % 10 = 5

## Hash table without needing dictionary

A white background with black and blue text

Description automatically generated

## Convert string to sorted array

Use sorted() method, which returns a char array with all the chars sorted

A screenshot of a computer program

Description automatically generated

## Place array elelemts and their count in hashmap one line

s = [‘A’, ‘B’,’C’,’D’]

hash\_table = collections.Counter(s)

A close-up of a website

Description automatically generated

## Enumerate loop

A computer code with colorful text

Description automatically generated

## Initialize empty 2d array 1 line

rows = [[0] \* N for \_ in range(N)]

