307. Range Sum Query - Mutable
We will use BIT to represent all sum numbers repectively. TC is O(nlogn), O(logn),O(logn)

```
class NumArray:

    def __init__(self, nums: List[int]):
        self.n = len(nums)
        self.a, self.c = nums, [0] * (self.n + 1)
        for i in range(self.n):
            k = i + 1
            while k <= self.n:
                self.c[k] += nums[i]
                k += (k & -k)

    def update(self, i: int, val: int) -> None:
        diff, self.a[i] = val - self.a[i], val
        i += 1
        while i <= self.n:
            self.c[i] += diff
            i += (i & -i)

    def sumRange(self, i: int, j: int) -> int:
        res, j = 0, j + 1
        while j:
            res += self.c[j]
            j -= (j & -j)
        while i:
            res -= self.c[i]
            i -= (i & -i)
        return res
```

94. Binary Tree Inorder Traversal
We will use stack to store all left nodes and pop out node to get val, then go through right branch tree. TC is O(n)

```
class Solution:
    def inorderTraversal(self, root: TreeNode) -> List[int]:
        res = []
        stack = []



        while stack or root:
            while root:
                stack.append(root)
                root = root.left
```

```
            root = stack.pop()
            res.append(root.val)
            root = root.right
        return res
```

589. N-ary Tree Preorder Traversal
We will use dfs to traverse all nodes. TC is O(n)

```
class Solution:
    def preorder(self, root: 'Node') -> List[int]:
        res = []
        if not root:
            return res
        res.append(root.val)
        for i in root.children:
            res.extend(self.preorder(i))
        return res
```

1217. Play with Chips
We only need to count odd and even numbers' count and return the smaller one.

```
class Solution:
    def minCostToMoveChips(self, chips: List[int]) -> int:
        odd, even = 0, 0
        for chip in chips:
            if chip % 2 == 0:
                even += 1
            else:
                odd += 1
        return odd if odd < even else even
```

1218. Longest Arithmetic Subsequence of Given Difference
We will use dp to get each current element's maximum pre subsequence. TC is O(n)

```
from collections import defaultdict
class Solution:
    def longestSubsequence(self, arr: List[int], difference: int) -> int:
        memo = defaultdict(int)
        for i in arr:
            memo[i] = max(memo[i - difference] + 1, memo[i])
        return max(memo.values())
```