695. Max Area of Island
We will use dfs to calculate all islands and reset that element to 0. TC is O(n)

```
class Solution:
    def maxAreaOfIsland(self, grid: List[List[int]]) -> int:
        if not grid or not grid[0]:
            return 0
        max_area = 0

        def helper(grid, i, j):
            if 0 <= i < len(grid) and 0 <= j < len(grid[0]) and grid[i][j]:
                grid[i][j] = 0
                return 1 + helper(grid, i + 1, j) + helper(grid, i - 1, j) + helper(grid, i, j - 1) + helper(grid,
i, j + 1)
            return 0

        for i in range(len(grid)):
            for j in range(len(grid[0])):
                if grid[i][j] == 1:
                    max_area = max(max_area, helper(grid, i, j))
        return max_area
```

240. Search a 2D Matrix II
We will start from left bottom, if the current one is larger than target, we move it to top, or we will move it to right. TC is O(n * m)

```
from bisect import *
class Solution:
    def searchMatrix(self, matrix, target):
        """
        :type matrix: List[List[int]]
        :type target: int
        :rtype: bool
        """
        if not matrix or not matrix[0]:
            return False

        row, col = len(matrix) - 1, 0
        while row >= 0 and col < len(matrix[0]):
            if matrix[row][col] == target:
                return True
            elif matrix[row][col] < target:
                col += 1
            else:
                row -= 1
```

```
        return False
```

234. Palindrome Linked List
We will reverse half linked list and compare one by one. TC is O(n), SC is O(1)

```
class Solution:
    def isPalindrome(self, head: ListNode) -> bool:
        dummy = ListNode(0)
        dummy.next = head
        dummy_mem = dummy
        if not head or not head.next:
            return True
        slow, fast = dummy.next, dummy.next.next
        while fast and fast.next:
            fast = fast.next.next
            node = slow.next
            slow.next = node.next
            node.next = dummy.next
            dummy.next = node

        dummy = dummy.next
        slow = slow.next
        if not fast:
            dummy = dummy.next
        while slow:
            if dummy.val != slow.val:
                return False
            dummy = dummy.next
            slow = slow.next
        return True
```

204. Count Primes
We will count all primes by multiplying all primes. TC is O(n)

```
class Solution:
    def countPrimes(self, n: int) -> int:
        memo = [False] * n
        count = 0
        for i in range(2, n):
            if memo[i]:
                continue
            count += 1
            for j in range(i * i, n, i):
                memo[j] = True
        return count
```

1048. Longest String Chain

We will use dp to add on largest value of words that missed one letter by 1. TC is O(n)

```
from collections import defaultdict
class Solution:
    def longestStrChain(self, words: List[str]) -> int:
        memo = defaultdict(int)
        for word in sorted(words, key=len):
            memo[word] = max([memo[word[:i] + word[i + 1:]] for i in range(len(word))]) + 1
        return max(memo.values())
```