1. 3 sum
   We will sort the whole array first, and then we will fix the first num and it downgrades a 2 sum question on the right side. We need to skip the element when the current element is the same as the previous one to prevent the duplication. TC is O(n^2)

```
class Solution:
    def threeSum(self, nums: List[int]) -> List[List[int]]:
        result = []
        if len(nums) < 3:
          return []
        nums.sort()

        for i in range(len(nums) - 2):
          l, r = i + 1, len(nums) - 1
          if i > 0 and nums[i] == nums[i - 1]:
            continue
          while l < r:
            if nums[l] + nums[r] == -nums[i]:
              result.append([nums[i], nums[l], nums[r]])
              l += 1
              while nums[l] == nums[l - 1] and l < r:
                l += 1
              r -= 1
              while nums[r] == nums[r + 1] and l < r:
                r -= 1
            elif nums[l] + nums[r] > -nums[i]:
              r -= 1
              while nums[r] == nums[r + 1] and l < r:
                r -= 1
            else:
              l += 1
              while nums[l] == nums[l - 1] and l < r:
                l += 1
        return result
```

2. Group Anagrams
   We will go through all strings in the array and store them in our dict, key is sorted string, value is the list of string. TC is O(length * n)

```
from collections import defaultdict
class Solution:
    def groupAnagrams(self, strs: List[str]) -> List[List[str]]:
        result = defaultdict(list)
        for s in strs:
          result[''.join(sorted(s))].append(s)
    return result.values()
```

3. Spiral Matrix
We will iterate from left to right, top to bottom, right to left, bottom to top, each time we finish one layer, we will add or reduce that layer by 1. Until left > right, or top > bottom. TC is O(m * n)

```
class Solution:
  def spiralOrder(self, matrix: List[List[int]]) -> List[int]:
    result = []
    if not matrix or not matrix[0]:
      return result
    left, right, top, bottom = 0, len(matrix[0]) - 1, 0, len(matrix) - 1

    while left <= right and top <= bottom:
      for i in range(left, right + 1):
        result.append(matrix[top][i])
      top += 1

      for i in range(top, bottom + 1):
        result.append(matrix[i][right])
      right -= 1

      if not left <= right or not top <= bottom:
        break

      for i in range(right, left - 1, -1):
        result.append(matrix[bottom][i])
      bottom -= 1

      for i in range(bottom, top - 1, -1):
        result.append(matrix[i][left])
      left += 1
    return result
```

4. Merge sorted array
We will sort our array from end to front until n < 0. TC is O(m + n)

```
class Solution:
  def merge(self, nums1: List[int], m: int, nums2: List[int], n: int) -> None:
    """
    Do not return anything, modify nums1 in-place instead.
    """
    i = m + n - 1
    m -= 1
```

```
        n -= 1
        while i >= 0:
          if m >= 0 and n >= 0:
            if nums1[m] > nums2[n]:
              nums1[i] = nums1[m]
              m -= 1
            else:
              nums1[i] = nums2[n]
              n -= 1
            i -= 1
          elif n >= 0:
            while n >= 0:
              nums1[i] = nums2[n]
              n -= 1
              i -= 1
          else:
            break
```
5. Pascal's Triangle

We will create layer by layer, we will create from 0 to i - 2 for each layer(i is the order of the layer). For each layer, except the first one, we will append 1 to the array's end and head. TC is O(n * n)

```
class Solution:
    def generate(self, numRows: int) -> List[List[int]]:
      result = []
      for i in range(1, numRows + 1):
        arr = [1]
        for j in range(i - 2):
          arr.append(result[-1][j] + result[-1][j + 1])
        if i > 1:
          arr.append(1)
        result.append(arr)
      return result
```