**[Group Anagrams](https://leetcode.com/problems/group-anagrams/)**

from collections import defaultdict

class Solution:

def groupAnagrams(self, strs):

anagram\_groups = defaultdict(list)

for s in strs:

key = ''.join(sorted(s))

anagram\_groups[key].append(s)

return list(anagram\_groups.values())

[**Destroying Asteroids**](https://leetcode.com/problems/destroying-asteroids/)

from typing import List

class Solution:

def asteroidsDestroyed(self, mass: int, asteroids: List[int]) -> bool:

asteroids.sort()

current\_mass = mass

for asteroid\_mass in asteroids:

if current\_mass < asteroid\_mass:

return False

current\_mass += asteroid\_mass

return True

[**Koko Eating Bananas**](https://leetcode.com/problems/koko-eating-bananas/)

from typing import List

class Solution:

def minEatingSpeed(self, piles: List[int], h: int) -> int:

l, r = 1, int(1e9)

while l < r:

mid = l + (r - l) // 2

hours = 0

for pile in piles:

hours += (pile + mid - 1) // mid

if hours <= h:

r = mid

else:

l = mid + 1

return l

[**Majority Element II**](https://leetcode.com/problems/majority-element-ii/)

class Solution:

    def majorityElement(self, nums: List[int]) -> List[int]:

                candidate1, candidate2 = None, None

        count1, count2 = 0, 0

        for num in nums:

            if num == candidate1:

                count1 += 1

            elif num == candidate2:

                count2 += 1

            elif count1 == 0:

                candidate1, count1 = num, 1

            elif count2 == 0:

                candidate2, count2 = num, 1

            else:

                count1 -= 1

                count2 -= 1

        return [m for m in (candidate1, candidate2) if nums.count(m) > len(nums) // 3]