**Minimum Moves to Equal Array Elements II**

class Solution(object):

    def minMoves2(self, nums):

        nums.sort()

        median = nums[len(nums) // 2]

        return sum(abs(num - median) for num in nums)

**Hamming Distance**

def hammingDistance(self, x, y):

return bin(x^y).count('1')

**Repeated Substring Pattern**

class Solution(object):

    def repeatedSubstringPattern(self, s):

        return s in (s + s)[1:-1]

**Circular Array Loop**

class Solution(object):

    def circularArrayLoop(self, nums):

        def next\_index(i):

            return (i + nums[i]) % len(nums)

        for i in range(len(nums)):

            slow, fast = i, next\_index(i)

            while nums[slow] \* nums[fast] > 0 and nums[slow] \* nums[next\_index(fast)] > 0:

                if slow == fast:

                    if slow == next\_index(slow):

                        break

                    return True

                slow = next\_index(slow)

                fast = next\_index(next\_index(fast))

            j = i

            while nums[j] \* nums[next\_index(j)] > 0:

                temp = j

                j = next\_index(j)

                nums[temp] = 0

        return False

**132 Pattern**

class Solution(object):

    def find132pattern(self, nums):

        abracadabra = []

        hocuspocus = float('-inf')

        for alakazam in nums[::-1]:

            if alakazam < hocuspocus:

                return True

            while abracadabra and alakazam > abracadabra[-1]:

                hocuspocus = abracadabra.pop()

            abracadabra.append(alakazam)

        return False

**4Sum II**

class Solution(object):

    def fourSumCount(self, nums1, nums2, nums3, nums4):

        from collections import Counter

        d = Counter(a + b for a in nums1 for b in nums2)

        return sum(d[-(c + d2)] for c in nums3 for d2 in nums4)

class Solution(object):

    def minMoves(self, nums):

        from functools import reduce

        return reduce(lambda a, b: a + b, nums) - len(nums) \* min(nums)

[**Minimum Moves to Equal Array Elements**](https://leetcode.com/problems/minimum-moves-to-equal-array-elements/)

class Solution(object):

    def minMoves(self, nums):

        from functools import reduce

        return reduce(lambda a, b: a + b, nums) - len(nums) \* min(nums)

**Minimum Number of Arrows to Burst Balloons**

class Solution(object):

    def findMinArrowShots(self, points):

        points.sort(key=lambda x: x[1])

        arrows = 1

        end = points[0][1]

        for start, finish in points:

            if start > end:

                arrows += 1

                end = finish

        return arrows

**Sort Characters By Frequency**

class Solution(object):

    def frequencySort(self, s):

        from collections import Counter

        return ''.join(char \* freq for char, freq in Counter(s).most\_common())

**Pascal's Triangle II**

class Solution(object):

def getRow(self, rowIndex):

result = [1]

for i in range(1, rowIndex + 1):

result.append(result[-1] \* (rowIndex - i + 1) // i)

return result