**Question 1** Given an integer array nums of 2n integers, group these integers into n pairs (a1, b1), (a2, b2),..., (an, bn) such that the sum of min(ai, bi) for all i is maximized. Return the maximized sum. **Example 1:** Input: nums = [1,4,3,2] Output: 4

class Solution:

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

nums.sort()

max\_sum=0

for i in range(0,len(nums),2):

max\_sum+=nums[i]

return max\_sum

**Question 2**

class Solution:

    def distributeCandies(self, candyType: List[int]) -> int:

        n = len(candyType)//2

        s = len(set(candyType))

        return min(n,s)

**Question 3**

class Solution:

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

freq = Counter(nums)

max\_length = 0

for key in freq:

if key + 1 in freq:

max\_length = max(max\_length, freq[key] + freq[key+1])

return max\_length

class Solution:

def canPlaceFlowers(self, flowerbed: List[int], n: int) -> bool:

f = [0] + flowerbed + [0]

for i in range(1,len(f)-1):

if f[i-1] == 0 and f[i] == 0 and f[i+1] == 0:

f[i] = 1

n -= 1

return n<=0

Q5

class Solution:

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

nums.sort()

h3=nums[-1]\*nums[-2]\*nums[-3]

l3=nums[0]\*nums[1]\*nums[-1]

return max(h3,l3)

Q6

class Solution:

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

        l = 0

        r = len(nums) - 1

        while l <= r:

            mid = (l + r) //2

            if nums[mid] == target:

                return mid

            elif nums[mid] > target:

                r = mid -1

            else:

                l = mid +1

        return -1

Q7

class Solution:

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

a = sorted(nums)

b = sorted(nums,reverse=True)

if nums == a or nums == b:

return True

return False

Q8

class Solution:

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

return max(0, max(nums) - min(nums) - 2 \* k)