**Assignment – 2**

Q1. 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

**Explanation:** All possible pairings (ignoring the ordering of elements) are:

1. (1, 4), (2, 3) -> min(1, 4) + min(2, 3) = 1 + 2 = 3
2. (1, 3), (2, 4) -> min(1, 3) + min(2, 4) = 1 + 2 = 3
3. (1, 2), (3, 4) -> min(1, 2) + min(3, 4) = 1 + 3 = 4 So the maximum possible sum is 4 <

Sol.

def arrayPairSum(nums):

nums.sort()

max\_sum = 0

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

max\_sum += nums[i]

return max\_sum

Q2. Alice has n candies, where the ith candy is of type candyType[i]. Alice noticed that she started to gain weight, so she visited a doctor. The doctor advised Alice to only eat n / 2 of the candies she has (n is always even). Alice likes her candies very much, and she wants to eat the maximum number of different types of candies while still following the doctor's advice. Given the integer array candyType of length n, return the maximum number of different types of candies she can eat if she only eats n / 2 of them. **Example 1:** Input: candyType = [1,1,2,2,3,3] Output: 3 **Explanation**: Alice can only eat 6 / 2 = 3 candies. Since there are only 3 types, she can eat one of each type.

Sol.

def distributeCandies(candyType):

unique\_types = len(set(candyType))

max\_candy\_types = min(unique\_types, len(candyType) // 2)

return max\_candy\_types

Q3. We define a harmonious array as an array where the difference between its maximum value and its minimum value is exactly 1. Given an integer array nums, return the length of its longest harmonious subsequence among all its possible subsequences. A subsequence of an array is a sequence that can be derived from the array by deleting some or no elements without changing the order of the remaining elements. **Example 1:** Input: nums = [1,3,2,2,5,2,3,7] Output: 5 **Explanation:** The longest harmonious subsequence is [3,2,2,2,3].

Sol. def findLHS(nums):

frequency = {}

for num in nums:

frequency[num] = frequency.get(num, 0) + 1

max\_length = 0

for num in set(nums):

if num + 1 in frequency:

max\_length = max(max\_length, frequency[num] + frequency[num + 1])

return max\_length

Q4. You have a long flowerbed in which some of the plots are planted, and some are not. However, flowers cannot be planted in adjacent plots. Given an integer array flowerbed containing 0's and 1's, where 0 means empty and 1 means not empty, and an integer n, return true if n new flowers can be planted in the flowerbed without violating the no-adjacent-flowers rule and false otherwise. **Example 1:** Input: flowerbed = [1,0,0,0,1], n = 1 Output: true

Sol.

def canPlaceFlowers(flowerbed, n):

count = 0

length = len(flowerbed)

for i in range(length):

if flowerbed[i] == 0 and (i == 0 or flowerbed[i - 1] == 0) and (i == length - 1 or flowerbed[i + 1] == 0):

flowerbed[i] = 1

count += 1

if count >= n:

return True

return False

Q5. Given an integer array nums, find three numbers whose product is maximum and return the maximum product. **Example 1:** Input: nums = [1,2,3] Output: 6

Sol.

def maximumProduct(nums):

nums.sort()

max\_product = nums[-1] \* nums[-2] \* nums[-3]

product\_with\_negatives = nums[0] \* nums[1] \* nums[-1]

return max(max\_product, product\_with\_negatives)