**Assignment-2**

**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

**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

Solution:

Algorithm:

1.sort the given array first.

2. Initiate a variable sum=0

3. Iterate over a loop with 2 step jump to get the min value.

CODE:

nums = [1,4,3,2]

def arraypair(nums : list[int]) -> int:

#sort the array

nums.sort()

#assign a sum variable

sum=0

#iterate over a for loop

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

sum += nums[i]

return sum

print(arraypair(nums))

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.

Solution:

Algorithm :

1. Get the size of candyType that alice can have i.e n= len(candyType)//2
2. We want distinct candies therefore apply set over the candyType to get the distinct values from it as ‘s’.
3. Return the min of n or s

CODE:

def diffcandies(candyType : list[int]) -> int:

n = len(candyType)//2

s = len(set(candyType))

return min(n,s)

print(diffcandies(candyType))

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].

Brute force: We can sort the array and then iterate over it till the time the difference is 1 between the numbers and if it changes then we can update or keep the “max” value as it is or according to the situation.

TC:- (nlogn), SC:- O(1)

Approach: Using the counting method or a Hashmap,

Algorithm: 1. Create a hashmap, it stores the count/frequency of each element in the map

2. Iterate over the map again and check for the key and see if any other key has a value one greater than the key value i.e key and (key+1)

3. Then we can return the count of both key values. Result = max(result, count(k)+count(k+1))

Solution:-

class solution(object):

def findlhs(self, nums: list[int])-> int:

nums = [1,3,2,2,5,2,3,7]

result = 0

hashmap = {}

numslen = len(nums)

for val in nums:

if val in hashmap:

hashmap[val]+=1

else:

hashmap[val] =1

inc = val +1

dec = val-1

if dec in hashmap:

result = max(result, hashmap[val]+hashmap[dec])

if inc in hashmap:

result = max(result, hashmap[val]+hashmap[inc])

return result

print(findlhs(nums))

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

Corner case:[0,0,1] => here though we have two zeros but we can still plant a flower on the 1st zero. [0,0,0]

Algorithm:

1. Lets assume two zero i.e empty flowerbed outside the given array. [0]+flowerbed+[0]
2. Now iterate over loop and check if i==0 and i+1 and i-1 ==0 if yes then we will decrement the value of given n
3. Now return n<=0 {it will return true or false}

Code:

def plantflower(flowerbed : list[int], n : int)-> bool:

f = [0] + flowerbed + [0]

for i in f(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. 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

Solution :-

Algorithm: 1. Sort the array and then return the last three nums products but this can even include the negative numbers therefore we will return the max from last three num product or 1st two and one last number.

Code:

nums = [1,2,3]

n = len(nums)

nums.sort()

print(nums)

return max(nums[n-1]\*nums[n-2]\*nums[n-3], nums[0]\*nums[1]\*nums[n-1])

Q6. Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search the target in nums. If target exists, then return its index. Otherwise, return -1.

You must write an algorithm with O(log n) runtime complexity.

Input: nums = [-1,0,3,5,9,12], target = 9 Output: 4

**Explanation:** 9 exists in nums and its index is 4

Solution: Approach – Binary search: O(logn)

Code:

def binarysearch(arr,i,j,x):

j= len(arr)-1

for i in range(0,len(arr)-1):

(i) + (j-i)//2 = mid

if mid == x:

return mid

elif mid < x:

return binarysearch(arr,mid+1,j,x)

elif mid>x:

return binarysearch(arr,i,mid-1,x)

else:

return -1

# Driver code

arr = [-1,0,3,5,9,12]

x = 9

i = 1

j = len(arr) - 1

# function calling

result = binarysearch(arr,i,j,x)

Q7. An array is monotonic if it is either monotone increasing or monotone decreasing. An array nums is monotone increasing if for all i <= j, nums[i] <= nums[j]. An array nums is monotone decreasing if for all i <= j, nums[i] >= nums[j]. Given an integer array nums, return true if the given array is monotonic, or false otherwise.

**Example 1:** Input: nums = [1,2,2,3] Output: true

Brute force: Traverse the array from left to right and then from right to left checking the values less than 1st index and last index value respectively if it satisfies in either of the case then return true.

**Algorithm**:

1. Traverse the array from index 0 to n-1{where n is the size of the array}
2. Check if arr[i]<=arr[i+1] for all the values in the array
3. Now check if arr[i]>=[i+1]

def check(arr):

N = len(arr)

inc = True

dec = True

for i in range(0, N-1):

if arr[i] > arr[i+1]:

inc = False

for i in range(0, N-1):

if arr[i] < arr[i+1]:

dec = False

return inc or dec

# Driver code

if \_\_name\_\_ == "\_\_main\_\_":

arr = [1, 2, 3, 3]

# Function call

ans = check(arr)

if ans == True:

print("Yes")

else:

print("No")

Q8. You are given an integer array nums and an integer k. In one operation, you can choose any index i where 0 <= i < nums.length and change nums[i] to nums[i] + x where x is an integer from the range [-k, k]. You can apply this operation at most once for each index i. The score of nums is the difference between the maximum and minimum elements in nums.

Return the minimum score of nums after applying the mentioned operation at most once for each index in it.

**Example 1:** Input: nums = [1], k = 0 Output: 0

**Explanation:** The score is max(nums) - min(nums) = 1 - 1 = 0.

Algorithm:

1. Take 2 variables min\_value and max\_value; min and max because we want to return the min score of nums which will be possible only after subtracting the max by given k and min value by adding the given k value.
2. Now after getting the min and max value from the loop we will return the value zero if the result is coming negative and if not then we will return the distance between the max and min value after subtracting and adding the k value.

def smallrange(arr:list[int], k : int):

min\_value = arr[0]

max\_value = arr[0]

for i in range(0,len(arr)-1):

min\_value = min(min\_value,arr[i])

max\_value = max(max\_value, arr[i])

if (min\_value + k) > (max\_value-k):

return 0

else:

return (max\_value-k) - (min\_value + k)

print(smallrange(arr,k))