Assignment 02 Solutions

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

In [17]:
```
def arrayPairSum(nums):
    nums.sort()
    sum_min = 0

    for i in range(0, len(nums), 2):
        sum_min += nums[i]

    return sum_min
```

In [18]:
```
nums = [1, 4, 3, 2]
print(arrayPairSum(nums))
```

```
4
```

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.

In [19]:
```
def distributeCandies(candyType):
    uniqueCandies = set()  # Step 1

    for candy in candyType:  # Step 2
        uniqueCandies.add(candy)

    limit = len(candyType) // 2  # Step 3
    return min(limit, len(uniqueCandies))  # Step 4
```

In [20]:
```
candyType = [1, 1, 2, 2, 3, 3]
print(distributeCandies(candyType))
```

```
3
```

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].

In [21]:
```
def findLHS(nums):
    numCount = {}
    for num in nums:
        numCount[num] = numCount.get(num, 0) + 1

    maxLength = 0
    for num in numCount.keys():
        if num + 1 in numCount:
            maxLength = max(maxLength, numCount[num] + numCount[num + 1])

    return maxLength
```

In [22]:
```
nums = [1, 3, 2, 2, 5, 2, 3, 7]
print(findLHS(nums))
```

5

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

In [23]:
```
def canPlaceFlowers(flowerbed, n):
    count = 0
    length = len(flowerbed)
    i = 0

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

    total = 0
    i = 0

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

        if total == n:
            return True

        i += 1

    return False
```

In [24]:
```
flowerbed = [1, 0, 0, 0, 1]
n = 1
print(canPlaceFlowers(flowerbed, n))  # Output: True
```

True

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

In [25]:
```
def maximumProduct(nums):
    nums.sort()
    return max(nums[-1] * nums[-2] * nums[-3], nums[0] * nums[1] * nums[-1])
```

In [26]:
```
nums = [1, 2, 3]
print(maximumProduct(nums))  # Output: 6
```

6

Q6. Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search 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

In [27]:
```
def search(nums, target):
    left, right = 0, len(nums) - 1

    while left <= right:
        mid = (left + right) // 2
```

```
        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return -1
```

In [28]:
```
nums = [-1, 0, 3, 5, 9, 12]
target = 9
print(search(nums, target))  # Output: 4
```

```
4
```

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

In [29]:
```
def isMonotonic(nums):
    isIncreasing = True
    isDecreasing = True

    for i in range(1, len(nums)):
        if nums[i] < nums[i - 1]:
            isIncreasing = False
        if nums[i] > nums[i - 1]:
            isDecreasing = False

    return isIncreasing or isDecreasing
```

In [30]:
```
nums = [1, 2, 2, 3]
print(isMonotonic(nums))  # Output: True
```

```
True
```

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.

In [31]:
```
def minimumScore(nums, k):
    minVal = min(nums)
    maxVal = max(nums)

    if k >= maxVal - minVal:
        return 0

    target = (minVal + maxVal) / 2

    for i in range(len(nums)):
        if nums[i] < target:
            nums[i] = minVal + k if target - nums[i] <= k else target
        else:
            nums[i] = maxVal - k if nums[i] - target <= k else target

    return max(nums) - min(nums)
```

In [32]:
```
nums = [1]
k = 0
print(minimumScore(nums, k))  # Output: 0
```

```
0
```

In [ ]: