DAA:-

Sorting Algorithm:-

1). GO COLORS

Solution:-

```
class Solution:
    def sortColors(self, nums: List[int]) -> None:
        N = len(nums)
        for i in range(N):
            flag = False
            for j in range(N - 1 - i):
                if nums[j] > nums[j + 1]:
                    flag = True
                    nums[j], nums[j + 1] = nums[j + 1], nums[j]
            if not flag:
                break
# Example usage:
nums = [2, 0, 2, 1, 1, 0]
solution = Solution()
solution.sortColors(nums)
print(nums)  # Output: [0, 0, 1, 1, 2, 2]
```

2). SORTING THE SEQUENCE

Solution:-

```
class Solution:
    def sortSentence(self, s: str) -> str:
        words = s.split()
        sorted_words = sorted(words, key=lambda word: int(word[-1]))
        reconstructed_sentence = ' '.join(word[:-1] for word in sorted_words)
        return reconstructed_sentence
```

```
# Example usage:
input1 = "is2 sentence4 This1 a3"
input2 = "Myself2 Me1 I4 and3"
solution = Solution()
output1 = solution.sortSentence(input1)
output2 = solution.sortSentence(input2)
print(f"Output for input 1: {output1}")  # Output: "This is a sentence"
print(f"Output for input 2: {output2}")  # Output: "Me Myself and I"
```

3). MINIMUM NUMBER GAME

Solution:-

```
from typing import List
class Solution:
    def numberGame(self, nums: List[int]) -> List[int]:
        nums.sort()
        arr = []
        left, right = 0, len(nums) - 1
        while left < right:
            arr.append(nums[right])
            right -= 1
            arr.append(nums[left])
            left += 1
        return arr
# Example usage:
sol = Solution()
nums = [2,5]
result = sol.numberGame(nums)
print(result)
```

4). AVARAGE SALARY EXCLUDING THE MINIMUM AND MXIMUM

Solution:-

```
from typing import List
class Solution:
    def average(self, salary: List[int]) -> float:
        def bubble_sort(nums: List[int]) -> List[int]:
            N = len(nums)
            for i in range(0, N):
                flag = False
                for j in range(0, N - 1 - i):
                    if nums[j] > nums[j + 1]:
                        nums[j], nums[j + 1] = nums[j + 1], nums[j]
                        flag = True
                if not flag:
                    break
            return nums
        sorted_salary = bubble_sort(salary.copy())
        total_amount = sum(sorted_salary[1:-1])
        N = len(salary)
        return total_amount / (N - 2)
# Example usage:
sol = Solution()
salary = [4000, 3000, 1000, 2000, 5000]
result = sol.average(salary)
print(result)
```

5). SORT EVEN AND ODD INDICES INDEPENDANTLY

Solution:-

```
class Solution:
    def sortEvenOdd(self, nums: list[int]) -> list[int]:
        n = len(nums)
        odd_indices = [nums[i] for i in range(1, n, 2)]
        even_indices = [nums[i] for i in range(0, n, 2)]
        odd_indices.sort(reverse=True)
        even_indices.sort()
        result = []
        odd_idx = 0
        even_idx = 0
        for i in range(n):
            if i % 2 == 0:
                result.append(even_indices[even_idx])
                even_idx += 1
            else:
                result.append(odd_indices[odd_idx])
                odd_idx += 1
        return result
# Example usage:
sol = Solution()
nums1 = [4, 1, 2, 3]
nums2 = [2, 1]
print(sol.sortEvenOdd(nums1))
print(sol.sortEvenOdd(nums2))
```

6). K WEAKEST ROWS IN A MATRICS

Solution:-

```
from typing import List, Tuple
class Solution:
    def kWeakestRows(self, mat: List[List[int]], k: int) -> List[int]:
        row_strength = []
        for idx, row in enumerate(mat):
            count_soldiers = sum(row)
            row_strength.append((count_soldiers, idx))
        row_strength.sort(key=lambda x: (x[0], x[1]))
        result = [row[1] for row in row_strength[:k]]
        return result
# Example usage:
sol = Solution()
mat1 = [
    [1, 1, 0, 0, 0],
    [1, 1, 1, 1, 0],
    [1, 0, 0, 0, 0],
    [1, 1, 0, 0, 0],
    [1, 1, 1, 1, 1]
]
k1 = 3
print(sol.kWeakestRows(mat1, k1))
mat2 = [
    [1, 0, 0, 0],
    [1, 1, 1, 1],
    [1, 0, 0, 0],
    [1, 0, 0, 0]
]
k2 = 2
print(sol.kWeakestRows(mat2, k2))
```

7). SQUARES OF A SORTED ARRAY

Solution:-

```
from typing import List
class Solution:
    def sortedSquares(self, nums: List[int]) -> List[int]:
        n = len(nums)
        squared_sorted = [0] * n
        left, right = 0, n - 1
        index = n - 1
        while left <= right:
            left_sq = nums[left] ** 2
            right_sq = nums[right] ** 2
            if left_sq > right_sq:
                squared_sorted[index] = left_sq
                left += 1
            else:
                squared_sorted[index] = right_sq
                right -= 1
            index -= 1
        return squared_sorted
# Example usage:
sol = Solution()
nums1 = [-4, -1, 0, 3, 10]
print(sol.sortedSquares(nums1))
nums2 = [-7, -3, 2, 3, 11]
print(sol.sortedSquares(nums2))
```

8). HEIGHT CHECKER

Solution:-

```
from typing import List
class Solution:
    def heightChecker(self, heights: List[int]) -> int:
        expected = sorted(heights)
        mismatch_count = 0
        for i in range(len(heights)):
            if heights[i] != expected[i]:
                mismatch_count += 1
        return mismatch_count
# Example usage:
sol = Solution()
heights1 = [1, 1, 4, 2, 1, 3]
print(sol.heightChecker(heights1))
heights2 = [5, 1, 2, 3, 4]
print(sol.heightChecker(heights2))
heights3 = [1, 2, 3, 4, 5]
print(sol.heightChecker(heights3))
```

9). RELATIVE RANKERS

Solution:-

```
from typing import List
class Solution:
    def findRelativeRanks(self, score: List[int]) -> List[str]:
        sorted_scores = sorted(score, reverse=True)
        rank_dict = {}
        for i in range(len(sorted_scores)):
            if i == 0:
                rank_dict[sorted_scores[i]] = "Gold Medal"
            elif i == 1:
                rank_dict[sorted_scores[i]] = "Silver Medal"
            elif i == 2:
                rank_dict[sorted_scores[i]] = "Bronze Medal"
            else:
                rank_dict[sorted_scores[i]] = str(i + 1)
        answer = [rank_dict[score[i]] for i in range(len(score))]
        return answer
# Example usage:
```

```
sol = Solution()
score1 = [5, 4, 3, 2, 1]
print(sol.findRelativeRanks(score1))
score2 = [10, 3, 8, 9, 4]
print(sol.findRelativeRanks(score2))
```

10). FIND TARGET INDICES AFTER SORTING ARRAY

Solution:-

```
def findTargetIndices(nums):
    indexed_nums = [(nums[i], i) for i in range(len(nums))]
    indexed_nums.sort()
    target_indices = [indexed_nums[i][1] for i in range(len(nums))]
    return target_indices
# Example usage:
nums = [5, 2, 6, 1]
result = findTargetIndices(nums)
print(result)
```

11). SPECIAL ARRAY WITH X ELEMENTS GREATER THAN OR EQUAL X

Solution:-

```
from typing import List
class Solution:
    def specialArray(self, nums: List[int]) -> int:
        nums.sort()
        n = len(nums)
        for x in range(n + 1):
            count = 0
            for num in nums:
                if num >= x:
                    count += 1
            if count == x:
                return x
        return -1
# Example usage:
sol = Solution()
print(sol.specialArray([3, 5]))
print(sol.specialArray([0, 0]))
print(sol.specialArray([0, 4, 3, 0, 4]))
```

12). BUY TWO CHOCOLATES

Solution:-

```
from typing import List
class Solution:
    def maxIceCream(self, prices: List[int], money: int) -> int:
        prices.sort()
        n = len(prices)
        min_money_left = float('inf')
        for i in range(n):
            for j in range(i + 1, n):
                total_cost = prices[i] + prices[j]
                if total_cost <= money:
                    min_money_left = min(min_money_left, money - total_cost)
        return min_money_left if min_money_left != float('inf') else money
# Example usage:
if __name__ == "__main__":
    solution = Solution()
    # Example 1
    prices1 = [1, 2, 2]
    money1 = 3
    print(f"Input: prices = {prices1}, money = {money1}")
    print("Output:", solution.maxIceCream(prices1, money1))
    # Example 2
    prices2 = [3, 2, 3]
    money2 = 3
    print(f"Input: prices = {prices2}, money = {money2}")
    print("Output:", solution.maxIceCream(prices2, money2))
```

13). HOW MANY NUMBERS ARE SMALLER THAN THE CURRENT

Solution:-

```
from typing import List
class Solution:
    def smallerNumbersThanCurrent(self, nums: List[int]) -> List[int]:
        count = [0] * 101
        for num in nums:
            count[num] += 1
        prefix_count = [0] * 101
        for i in range(1, 101):
            prefix_count[i] = prefix_count[i - 1] + count[i - 1]
        result = []
        for num in nums:
            result.append(prefix_count[num])
        return result
# Example usage:
if __name__ == "__main__":
    solution = Solution()
    # Example 1
    nums1 = [8, 1, 2, 2, 3]
    print(f"Input: nums = {nums1}")
    print("Output:", solution.smallerNumbersThanCurrent(nums1))
    # Example 2
    nums2 = [6, 5, 4, 8]
    print(f"Input: nums = {nums2}")
    print("Output:", solution.smallerNumbersThanCurrent(nums2))
    # Example 3
    nums3 = [7, 7, 7, 7]
    print(f"Input: nums = {nums3}")
    print("Output:", solution.smallerNumbersThanCurrent(nums3))
```

14). SORT ARRAY BY INCREASING FREQUENCY

Solution:-

```
from collections import Counter
class Solution:
    def frequencySort(self, nums):
        freq = Counter(nums)
        sorted_nums = sorted(nums, key=lambda x: (freq[x], -x))
        return sorted_nums
# Examples from the prompt:
nums1 = [1, 1, 2, 2, 2, 3]
nums2 = [2, 3, 1, 3, 2]
nums3 = [-1, 1, -6, 4, 5, -6, 1, 4, 1]
sol = Solution()
print(sol.frequencySort(nums1))  # Output: [3, 1, 1, 2, 2, 2]
print(sol.frequencySort(nums2))  # Output: [1, 3, 3, 2, 2]
print(sol.frequencySort(nums3))  # Output: [5, -1, 4, 4, -6, -6, 1, 1, 1]
```

15). SET MISMATCH

Solution:-

```
class Solution:
    def findErrorNums(self, nums):
        n = len(nums)
        actual_sum = sum(nums)
        expected_sum = n * (n + 1) // 2
        missing_num = expected_sum - actual_sum
        seen = set()
        duplicated_num = None
        for num in nums:
            if num in seen:
                duplicated_num = num
                break
            seen.add(num)
        return [duplicated_num, duplicated_num + missing_num]
sol = Solution()
# Example 1
nums1 = [1, 2, 2, 4]
print(sol.findErrorNums(nums1))
# Example 2
nums2 = [1, 1]
print(sol.findErrorNums(nums2))
```

16). FIND ALL NUMBERS DISAPPEARD IN ARRAY
Solution:-

```
    class Solution:
        def findDisappearedNumbers(self, nums: list[int]) -> list[int]:
            for num in nums:
                index = abs(num) - 1
                nums[index] = -abs(nums[index])
            result = [i + 1 for i, num in enumerate(nums) if num > 0]
            return result
    # Example usage:
    sol = Solution()
    print(sol.findDisappearedNumbers([4, 3, 2, 7, 8, 2, 3, 1]))
    print(sol.findDisappearedNumbers([1, 1]))
```

17). FIND THE DUPLICATE NUMBER
Solution:-

```
class Solution:
    def findDuplicate(self, nums: list[int]) -> int:
        slow, fast = nums[0], nums[0]
        while True:
            slow = nums[slow]
            fast = nums[nums[fast]]
            if slow == fast:
                break
        slow = nums[0]
        while slow != fast:
            slow = nums[slow]
            fast = nums[fast]
        return slow
# Example usage:
sol = Solution()
print(sol.findDuplicate([1, 3, 4, 2, 2]))
print(sol.findDuplicate([3, 1, 3, 4, 2]))
print(sol.findDuplicate([3, 3, 3, 3, 3]))
```

18). FIND ALL DUPLICATE IN AN ARRAY

Solution:-

```
class Solution:
    def findDuplicates(self, nums: List[int]) -> List[int]:
        result = []
        for num in nums:
            index = abs(num) - 1
            if nums[index] < 0:
                result.append(index + 1)
            else:
                nums[index] = -nums[index]
        return result
```

```
# Example usage:
sol = Solution()
print(sol.findDuplicates([4, 3, 2, 7, 8, 2, 3, 1]))
print(sol.findDuplicates([1, 1, 2]))
print(sol.findDuplicates([1]))
```

19). MISSING NUMBER

Silution:-

```
class Solution:
    def missingNumber(self, nums: List[int]) -> int:
        n = len(nums)
        total_sum = n * (n + 1) // 2
        actual_sum = sum(nums)
        return total_sum - actual_sum
# Test cases
solution = Solution()
nums1 = [3, 0, 1]
print(solution.missingNumber(nums1))
nums2 = [0, 1]
print(solution.missingNumber(nums2))
nums3 = [9, 6, 4, 2, 3, 5, 7, 0, 1]
print(solution.missingNumber(nums3))
```