Day_17:DSA

Day-17 : Leetcode: 2033 Minimum Operations to Make a Uni-Value - Grid.

Ex.

2	4
6	8

m guid

grid = [[2,4],[6,8]] , x=2 } goal is to make every value in grid same and we have to return minimum no. of operations.

Output = 4

Code:

```
Class Solution:
    def minOperations(self, grid:)
        total = 0
        for row in grid:
            for n in row:
                if n % x != grid[0][0] % x:
                    return -1
        num = [n for row in grid for n in row]
        nums.sort()
        prefix = 0
        res = float("inf")
        for i in range(len(nums)):
            cost_left = nums[i] * i - prefix
            cost_right = total - prefix - (nums[i] * (len(nums)-i))
            operations = (cost_left + cost_right)//x
            res = min(res, operations)
            prefix += nums[i]
        return res
```

SDE: . Repeat and Missing Number

- Brute force:
  - We will run a loop (say i) from 1 to N
  - For each integer, i we will count occurrence in the given array using linear search
  - We will store those two elements that has the occurrence of 2 and 0
  - Finally we will return.

Code:

```
def FindmissingRepeatingNumber (a: [int]) -
    n = len(a)
    repeating, missing = -1, -1

    for i in range (1, n+1):
        cnt = 0
        for j in range(n):                    TC : O(N^2)
            if a[j] == i:
                cnt += 1

        if cnt == 2:
            repeating = i
        elseif cnt == 0:
            missing = i
        if repeating != -1 and missing != -1:
            break

    return [repeating, missing]
```

- Better: (Hashing)

```
def find ----
    n = len(a)
    hash = [0] * (n+1)                    T.C = O(N)
    for i in range(n):                    S.C = O(N)
        hash[a[i]] += 1

    repeating, missing = -1, -1
    for i in range(1, n+1):
        if hash[i] == 2:
            repeating = i
        elseif hash[i] == 0
            missing = i
```

• Optimal Soln (Using Maths)

```
def FindMissing ----
    n = len(a)

    # Find the sum
    SN = (n * (n+1)) // 2
    S2N = (n * (n+1) * (2 * n+1)) // 6

    # Calculate Sand S2
    S, S2 = 0, 0
    for i in range(n):
        S += a[i]
        S2 += a[i] * a[i]

    Val1 = S - SN
    Val2 = S2 - S2N

    Val2 = Val2 // val1

    x = (val1 + val2) // 2
    y = x - Val1
    return [x, y]
```

T.C → O(N)
S.C → O(1)

Find the duplicate in an array of N+1 integers

→ Assuming there is one dupe

arr = [1, 3, 4, 2, 2]

output: 2.

- Using Sorting

```
if (arr[i] == arr[i+1]) {
    return arr[i];
}
}
```

- Using frequency

```
n = len(arr)
freq = [0] * (n+1)
for i in range
    if freq[arr[i]] == 0;
        freq[arr[i]] += 1
    else:
        return arr[i]
return 0
```

- def findDuplicate(nums: List[int]) -> int

```
slow = nums[0]
fast = nums[0]
while True:
    slow = nums[slow]
    fast = nums[nums[fast]]
    if slow == fast:
        break
fast = nums[0]
while slow != fast:
    slow = nums[slow]
    fast = nums[fast]
return slow
```