Day-7

Python DSA

31 Next Permutation

https://leetcode.com/problems/next-permutation/

Optimal Solution

```
class Solution:
    def nextPermutation(self, nums: List[int]) -> None:
        n = len(nums)

        ind = -1
        for i in range(n - 2, -1, -1):
            if nums[i] < nums[i + 1]:
                ind = i
                break

        if ind == -1:
            nums.reverse()
            return nums

        # Step 2: Find the next greater element
        #         and swap it with nums[ind]:
        for i in range(n - 1, ind, -1):
            if nums[i] > nums[ind]:
                nums[i], nums[ind] = nums[ind], nums[i]
                break

        # Step 3: reverse the right half:
        nums[ind + 1 :] = reversed(nums[ind + 1 :])
```

Dry Run
Example Input: nums = [1, 2, 3]
Step 1: Find the first decreasing element from the right
We traverse from the end of the array to the beginning to find the first pair nums[i] < nums[i+1].

Start from i = n-2 = 1:
nums[1] = 2, nums[2] = 3 → 2 < 3 → Condition satisfied
Set index: ind = 1
Result after Step 1:

ind = 1
Step 2: Find the element just larger than nums[ind] and swap
Now we find the smallest element on the right of ind that is greater than nums[ind].

Start from i = n-1 = 2:
nums[2] = 3 > nums[1] = 2 → Condition satisfied
Swap nums[1] and nums[2]
Array after swapping:

Before swap: [1, 2, 3]
After swap: [1, 3, 2]
Step 3: Reverse the subarray to the right of ind
Finally, we reverse the elements from ind + 1 to the end of the array to get the smallest possible permutation of the remaining elements.

Reverse nums[2:] → [2] → remains [2]
Final Output:

[1, 3, 2]

Edge Case Dry Run: nums = [3, 2, 1]
Step 1: Find the first decreasing element from the right

Check from right:
nums[1] = 2, nums[2] = 1 → 2 > 1 ❌
nums[0] = 3, nums[1] = 2 → 3 > 2 ❌
No such element found → ind = -1
Step 2: No valid index found

Since ind = -1, reverse the entire array
Before reverse: [3, 2, 1]

After reverse: [1, 2, 3]

Time Complexity- O(3N)

Space Complexity – O(1)

73 Set Matrix Zeros

https://leetcode.com/problems/set-matrix-zeroes/

Optimal Solution

```
class Solution:
    def setZeroes(self, matrix: List[List[int]]) -> None:
        """
        Do not return anything, modify matrix in-place instead.
        """
        r= len(matrix)
        c= len(matrix[0])

        col0=1
        for i in range(r):
            for j in range(c):
                if matrix[i][j]==0:
                    if j==0:
                        col0=0
                    else:
                        matrix[0][j]=0
                        matrix[i][0]=0

        for i in range(1,r):
            for j in range(1,c):
                if matrix[0][j]==0 or matrix[i][0]==0:
                    matrix[i][j]=0
        for j in range(c-1,0,-1):
            if matrix[0][0]==0:
                matrix[0][j]=0
        for i in range(0,r):
```

```
        if col0==0:
            matrix[i][0]=0
```

Dry Run: setZeroes Function

Problem: Set entire row and column to 0 if any element in the matrix is 0.
Approach: In-place using the first row and first column as markers.

Example Input:

```
matrix = [
   [1, 1, 1],
   [1, 0, 1],
   [1, 1, 1]
]
```

Step 1: Initialize Variables

* r = 3 (number of rows)
* c = 3 (number of columns)
* col0 = 1 (to track whether first column needs to be zeroed)

Step 2: First Pass — Mark the rows and columns

Loop through the matrix to find zeros and mark the corresponding first row and first column:

* i = 0, no 0 in row → do nothing
* i = 1, j = 1 is 0 →
  → mark `matrix[0][1] = 0`, mark `matrix[1][0] = 0
* `i = 2`, no 0 in row → do nothing

Matrix after marking:

1 0 1

0 0 1
1 1 1

Step 3: Second Pass — Update cells using the markers

Loop from i = 1 and j = 1 to c - 1:

* i = 1, j = 1: Already 0 → stays 0
* i = 1, j = 2: `matrix[1][0] == 0` → set matrix[1][2] = 0
* i = 2, j = 1: `matrix[0][1] == 0` → set matrix[2][1] = 0
* i = 2, j = 2: no marker → remains unchanged

Matrix after update:

1 0 1
0 0 0
1 0 1

Step 4: Handle first row

Check if matrix[0][0] == 0. It's not (matrix[0][0] == 1)
→ So skip zeroing the first row.

Step 5: Handle first column

Since col0 == 1 → First column is not modified.

Final Output:

1 0 1
0 0 0
1 0 1

Time Complexity – O(M*N)

Space Complexity- O(1)