Python DSA

Day –2

1. Leetcode 189 Rotate Array

https://leetcode.com/problems/rotate-array/description/

Solution

Optimal Solution

```
class Solution:
    def rotate(self, nums: List[int], k: int) -> None:
        """
        Do not return anything, modify nums in-place instead.
        """
        def reverse(l,r):
            while l < r:
                nums[l], nums[r]=nums[r], nums[l]
                l +=1
                r -=1
        n = len(nums)
        k %=n
        reverse(n-k, n-1)
        reverse(0,n-k-1)
        reverse(0,n-1)
```

Dry Run

nums = [1, 2, 3, 4, 5, 6, 7], k = 3

Expected Output = [5, 6, 7, 1, 2, 3, 4]

Step –1 Initial Setup

nums = [1, 2, 3, 4, 5, 6, 7]

n = 7

k = 3 → No change because 3 < 7

Step –1 Reverse the Last K Elements

Before: [1, 2, 3, 4, 5, 6, 7]

Index: ↑   ↑

              n-k=4   n-1=6

Reverse [5, 6, 7] → [7, 6, 5]

After: [1, 2, 3, 4, 7, 6, 5]

Step –2 Reverse the first n-k elements

Reverse the first 4 elements

Before: [1, 2, 3, 4, 7, 6, 5]

Index: ↑   ↑

                 0   n-k-1=3

Reverse [1, 2, 3, 4] → [4, 3, 2, 1]

After: [4, 3, 2, 1, 7, 6, 5]

Step –3 Reverse the Entire Array

Before: [4, 3, 2, 1, 7, 6, 5]

Index: ↑   ↑

               0   6

Reverse whole array → [5, 6, 7, 1, 2, 3, 4]

Time Complexity = O(n)

Space Complexity = O(1)

2. Leetcode 283 Move Zeros to End

https://leetcode.com/problems/move-zeroes/

Optimal Solution

```
class Solution:
    def moveZeroes(self, nums: List[int]) -> None:
        """
        Do not return anything, modify nums in-place instead.
        """
        if len(nums)==1:
            return
        i=0
        while i < len(nums):
            if nums[i] ==0:
                break
            i +=1
        if i == len(nums):
            return
        j = i+1
        while j < len(nums):
            if nums[j] !=0:
                nums[i] , nums[j] = nums[j] , nums[i]
                i +=1
            j +=1
```

Dry Run

Input:

```
nums = [0, 1, 0, 3, 12]
```

Step 1: Find the first zero

- Start with `i = 0`
- `nums[0] == 0`, so we stop here
- `i = 0` (points to the first zero)

Step 2: Move non-zero elements forward

Start with `j = i + 1 = 1`

- `j = 1`, `nums[1] = 1` → not zero

Swap `nums[0]` and `nums[1]` → `[1, 0, 0, 3, 12]`

Increment `i → 1`

- `j = 2`, `nums[2] = 0` → skip

No change

- `j = 3`, `nums[3] = 3` → not zero

Swap `nums[1]` and `nums[3]` → `[1, 3, 0, 0, 12]`

Increment `i → 2`

- `j = 4`, `nums[4] = 12` → not zero

Swap `nums[2]` and `nums[4]` → `[1, 3, 12, 0, 0]`

Increment `i → 3`

Final Output:

```
nums = [1, 3, 12, 0, 0]
```

All non-zero elements are moved to the front in order, and zeros are moved to the end.

Time Complexity – O(n)

Space Complexity – O(1)

3. Array Search

https://www.geeksforgeeks.org/problems/search-an-element-in-an-array-1587115621/1

```
class Solution:
    def search(self, arr, x):
        for i in range(0, len(arr)):
            if arr[i] == x:
                return i
        return –1
```

Time Complexity – O(n)

Space Complexity- O(1)

4. Union of 2 Sorted Arrays

https://www.geeksforgeeks.org/problems/union-of-two-sorted-arrays-1587115621/1

Optimal Solution

```
class Solution:
    def findUnion(self, a, b):
        i = 0
        j = 0
        result = []
        while i < len(a) and j < len(b):
            if a[i] <= b[j]:
                if len(result) == 0 or result[-1] != a[i]:
                    result.append(a[i])
                i += 1
            else:
                if len(result) == 0 or result[-1] != b[j]:
                    result.append(b[j])
                j += 1

        while i < len(a):
            if len(result) == 0 or result[-1] != a[i]:
                result.append(a[i])
            i += 1

        while j < len(b):
            if len(result) == 0 or result[-1] != b[j]:
                result.append(b[j])
            j += 1

        return result
```

Dry Run

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

b = [2, 3, 5, 6]

i = 0, j = 0

result = []

While Loop 1: `while i < len(a) and j < len(b)`

i	j	a[i]	b[j]	result before	Action	result after
0	0	1	2	[]	1 <= 2 → append 1 from a	[1]
1	0	2	2	[1]	2 <= 2 → append 2 from a	[1, 2]
2	0	2	2	[1, 2]	2 == 2 but already in result	skip
3	0	3	2	[1, 2]	3 > 2 → append 2 from b	[1, 2] (already added, skip)
3	1	3	3	[1, 2]	3 == 3 → append 3 from a	[1, 2, 3]
4	1	4	3	[1, 2, 3]	4 > 3 → skip b[1] (already in res)	
4	2	4	5	[1, 2, 3]	4 <= 5 → append 4 from a	[1, 2, 3, 4]

Now `i == len(a)`, exit this loop.

While Loop 2: `while i < len(a)`

- Skipped, because `i == len(a)`

While Loop 3: `while j < len(b)`

j	b[j]	result before	Action	result after
2	5	[1, 2, 3, 4]	append 5	[1, 2, 3, 4, 5]
3	6	[1, 2, 3, 4, 5]	append 6	[1, 2, 3, 4, 5, 6]

Output

[1, 2, 3, 4, 5, 6]

Time Complexity – O(m+n)

Space Complexity – O(1)