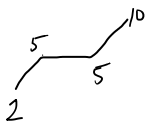
Day - 3

Saturday, 27 December 2025 11:11 AM

Q. Given an arr. of size 'n'. check if the given arr is sorted or not?

asc ↑

eg [1, 2, 5, 5, 10]



op : True

eg [2, 5, 4, 10]

op : False

```
isSorted(nums[], n) {
    for(i=0; i<n-1; i++) {
        if(nums[i] > nums[i+1])
            return false.
    }
    return True
}
```

[1, 2, 5, 5, 10] → > → false.

i = [n-1] > [n]

TC = O(n)

SC = O(1)

Q Given an arr of size 'n'. Reverse it.

eg [1, 7, 3, 10]

o/p ⇒ [10, 3, 7, 1]

l	r
0	3 (0<3) swap ✓
1	2 (1<2) swap ✓
2	1 (2<1) stop

```
reverse(nums[], n) {
    left = 0 ; right = n-1;
    while(left ≤ right) {
        int temp = nums[left];
        nums[left] = nums[right]
        nums[right] = temp;
        left++;
        right--
    }
}
```

TC = O(n/2) = O(n) < O(n)

SC = O(1)

Q Given an arr of size `n` and `2` end-points

Just reverse the sub-arr within the end-points.

eg [1, 2, 5, 9, 7, 12, -3] u=2 v=5 (u ≤ v)

0 1 2 3 4 5 6

o/p: [1, 2, 12, 7, 9, 5, -3]

```
reverseInRange(nums[], n, u, v){
    left = u; right = v;
    while(left ≤ right){
        int temp = nums[left];
        nums[left] = nums[right]
        nums[right] = temp;
        left++;
        right--
    }
}
```

TC = O(v-u) → O(n) → W.C

SC = O(1)

Q Rotate an arr of size `n` by "k" - steps. [k > n]

eg [1, 2, 3, 4, 5] k=2

1st Rotn → [5, 1, 2, 3, 4]

2nd Rotn [4, 5, 1, 2, 3] ✓

o/p

eg [1, 5, 7, 4] k=3

o/p [5, 7, 4, 1]

k = k%n ← [0, n-1]

num = | 1 | 5 | 7 | 4 | (0 1 2 3)

temp ⇒ | 5 | 7 | 4 | 1 | (0 1 2 3)

0+3 = 3%4 = 3

1+3 = 4%4 = 0

2+3 = 5%4 = 1

3+3 = 6%4 = 2

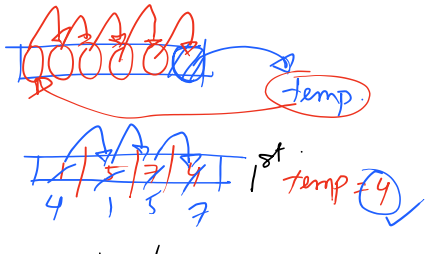
M-1 Bruteforce

K = K%N

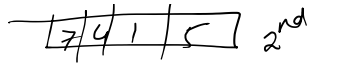
```
for(i=1; i≤k; i++){
    temp = nums[n-1];
    for(j=n-2; j>=0; j--){
        nums[j+1] = nums[j]
```

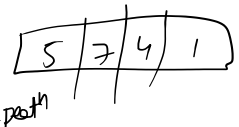
k-times

O(n)

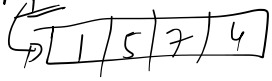


num[0] = temp





Repeat

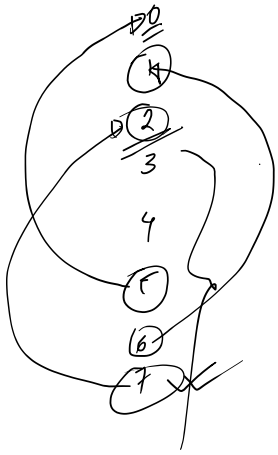


n=5

K = 7

k = 7 % 5

= 2



TC = K · O(n) = O(n·k)

SC = O(1)

M-2 "Extra-Space"

```
temp[n]
for (i=0; i ≤ n-1; i++) {
    targetIdx = (i+k) % n
    temp[targetIdx] = nums[i];
}
for i ← 0 .. n-1 {
    nums[i] = temp[i]
}
```

O(n·k)

TC = O(n)

SC = O(n)

O(1)

M-3

[1, 2, 3, 5, 7, 4] (S1) ⇒ [(4, 7, 5) (3, 2, 1)]

// (S-2)

[5, 7, 4) (1, 2, 3] <=> [5, 7, 4, 1, 2, 3]

//

[4, 7, 5, 3, 2, 1]

reverseInRange(nums, 0, n-1); ⇒ O(n)

reverseInRange(nums, 0, k-1); ⇒ O(n)

reverseInRange(nums, k, n-1); ⇒ O(n)

⇒ O(n)

TC → O(n)

SC → O(1)

Q. Generate all the sub-arrays.

eg: [1, 4, 2] ⇒

o/p [1] [1,4] [1,4,2]

[4] [4,2]

[2]

[1,4,2] (may not be contiguous)

sub-arr	sub-seq
[1,4] ✓	[1,4] ✓
[1,2] ✗	[1,2] ✓
[1] ✓	[1] ✓
[2,1] ✗	[2,1] ✗
[4,1] ✗	[4,1] ✗
[4,2] ✓	[4,2] ✓

given an arr of size "N" ⇒ $\frac{n(N+1)}{2}$

how many sub-arrs will be there? non-empty

eg: n=1 [1] ⇒ [1] ✓ ⇒ 1

n=2 [1,2] ⇒ [1] [1,2] [2] ⇒ 3 (1+2)

⇒ 6 (1+2+3)

n=3

n=4 [1,2,3,4] ⇒ [1] [1,2] [1,2,3]
[4] [2,3,4]
[2] [2,3] [3,4]
[3] [4]

⇒ 10 (1+2+3+4)

n=5 ⇒ $\frac{n(n+1)}{2}$ ⇒ $\frac{5 \times 6}{2}$ ⇒ 15 ✓

given an arr of size "N" ⇒ $(2^N - 1)$

how many non-empty sub-seq will be there?

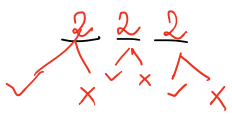
eg n=1 [1] ⇒ [1] ⇒ $1 (2^1 - 1)$

n=2 [1,2] ⇒ [1], [2], [1,2] ⇒ 3 (2^2-1)

n=3 [1,2,3] ⇒ [1] [2] [3] ✓
[1,2] [1,3] [2,3] ✓ ⇒ 7 (2^3-1)
[1, 2, 3]

n⇒4 [1,2,3,4) ⇒ [1), [2], [3), [4) ⇒ 4
[1,2] [1,3) [1,4) ⇒ 6
[2,3] [2,4] [3,4)
[1,2,5] [1,2,4) ⇒ 4
[2,3,4] [1,3,4]
[1,2,3,4) ⇒ 1

⇒ 15 (2^4-1)



⇒ 2·2·2 ⇒ $[2^3-1]$

[[—), [
ans = [ ]
for (i = 0; i < n; i++) {
temp = [ ] ✓
for (j = i; j < n; j++) {
temp.append (nums[j])
ans.append (temp)
}
}

temp
[1,2,3]
temp
ans [[1], [1,2], [1,2,3] ...]
ans [[1]]
[[1], [1,2]

e
[1, 2, 3, 4]
r
ans = [ ]
for l ← 0 ... n-1 {
for r ← l ... n-1

```
temp=[]
for i in (l...r) {
    temp.append(nums[i])
}
ans.append(temp)
}
```

=> $O(n^3)$