Day - 5

Sunday, 4 January 2026 11:01 AM

Q. Given an arr. Find largest no.

eg [2, 8, 1, 11, -5] o/p ⇒ 11 ✓

m-1

```
maxVal = Int_Min (-∞)
for (i=0; i<n; i++) {
    if (maxVal < arr[i])
        maxVal;
}
maxVal;
```

⇒ TC = O(n)

SC = O(1)

m-2 — sort ↓ asc ↳ of arr[n-1]

TC = O(n log n)

SC = O(1)

Q. Given an arr. Find 2nd largest no.

eg [2, 8, 1, 11, -5] o/p ⇒ 8 ✓

m-1

```
max1 = -∞   max2 = -∞
for (i=0; i<n; i++) {
    if (arr[i] > max1)
        max2 = max1
        max1 = arr[i]
    else if (arr[i] > max2)
        max2 = arr[i]
}
```

TC = O(n)

SC = O(1)

m-2 ⇒ Sort it (arr) ↳ [n-2] ✓

[1, 3, 2]

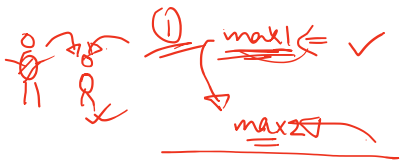
max1 = -∞

max2 = -∞

i	max1	max2
0	~~-∞~~ 1	-∞
1	~~1~~ 3	~~-∞~~ 1
2	3	2

[1, 2, 3]

i	max1	max2
0	~~-∞~~ 1	-∞
1	* 2	-∞
2	~~2~~ 3 ✓	-∞



Q Rt ⇒ 3rd highest

Paypal

Q Move zeros to end

eg. [2, 1, 0, 3, 0, 0, 5, 9]

l r

s e

[2, 0, 3, 0, 1]

o/p [2, 1, 3, 5, 9, 0, 0, 0]

[2, 0, 3, 0, 1]
1 0

l=0 ; r=n-1

while (l<r) {

aaaaaa
aaaaaa

while (l<r and arr[l] != 0) l++;

while (l<r and arr[r] == 0) r--;

if (l<r) {

swap(arr[l], arr[r])

l++

r--

}

}

TC=O(n)
SC=O(1)

Amazon

Q Sort an arr. of 0's, 1's and 2's.

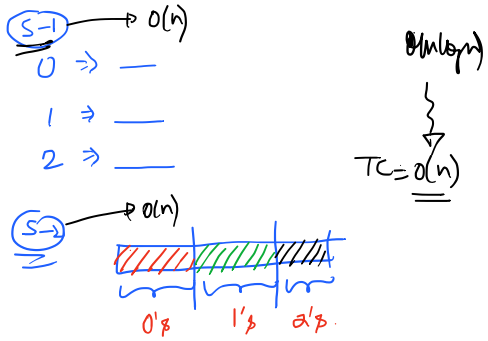
eg [0,2,1,0,0,2,1] → o/p

[0,0,0,1,1,2,2]

M-1 ⇒ sort ⇒ TC=O(nlogn)

SC⇒ O(1)

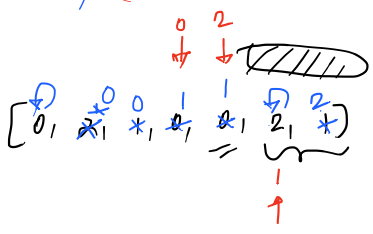
M-2 ⇒



O(n+n)

TC=O(n)

M-3 (Three Pointers) (DNF Algo)



i=0 j=0 k=n-1

while (j ≤ k) {

if (arr[j] == 0) {

swap(arr[i], arr[j])

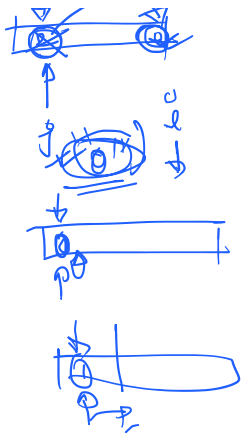
} i++; j++

i

1

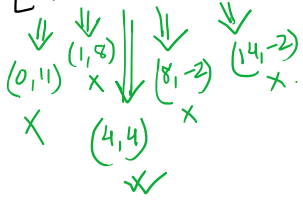
```
else if (a[j] == -1) {
    j++
}
else {
    swap(a[j], a[k])
    k--;
}
```

TC = O(n)
SC = O(1)



Q. Equilibrium Point (LM) (find an idx s.t. sum of all the left elem = sum of all right elem.

eg [1, 3, 4, 6, -2]



op ⇒ (2) ✓

eg [5]

op ⇒ (0) ✓

eg [5, 6]

op ⇒ ✗ (-1) ✓

eg [6, 6] ⇒ (-1) ✓

⇓ ⇔ (6, 0)

(0, 6)

eg [0, 2, 2, 0]

op ⇒ (1) 2 ⇒

⇓

M-1 Naive-Approach

```
for (i ← 0; i < N; i++) {
    Lsum = 0   Rsum = 0
    for (j = 0; j < i; j++)
        Lsum += arr[j]
    for (j = i+1; j < n; j++)
        Rsum += arr[j]
    if (Lsum == Rsum)
        return i;
}
return -1;
```

TC = O(n²)
SC = O(1)

M-2

```
TotalSum = 0
for (i = 0; i < n; i++) {
    totalSum += arr[i]
```

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

```
}
curSum = 0
for (i = 0; i < n; i++) {
    remSum = TotalSum - curSum - arr[i]
    if (remSum == curSum)
        ans = i
    curSum += arr[i]
}
```

i	totalSum	curSum	RemSum
0 (1)	12	0 → 1 ①	12-0-1 = ⑪
1 (5)	12	1 → 3 ④	12-1-3 = ⑧
2 (4)	12	4	12-4-4 = ④ ✓

TC = O(n)

SC = O(1)

Q Max Sub-array Sum

eg [2, 1, -2, 5, -10] ans ⇒ 6 ✓

[5] ⇒ 5 [2, 1, -2, 5] ⇒ 6 ✗

[2, 1] ⇒ 3

eg [2, 1, 2) → +ve

$\sum_{i=0}^{n-1} A[i]$

M-1 Naive App

```
maxVal = -∞
for (i = 0; i < n; i++) {
    curSum = 0
    for (j = i; j < n; j++) {
        curSum += arr[j]
        if (curSum > maxV)
            maxVal = curSum
    }
}
```

[2, 1, -2, 5, -10] (i, j)

curSum = ~~0~~ ~~2~~ ~~3~~ ~~4~~ ~~6~~ -4

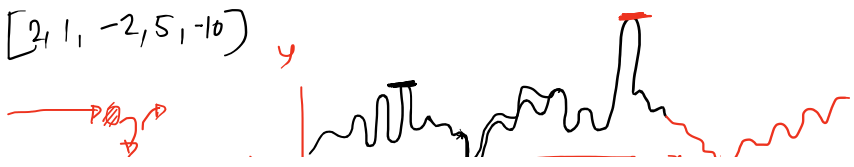
maxV = ~~2~~ ~~3~~ ~~4~~ 6

TC = O(n²)

SC = O(1)

M-2

[2, 1, -2, 5, -10] (s, e)



(0,0)

```
maxVal = -∞
currSum = 0
for (i = 0; i < n; i++) {
    currSum += a[i]
    if (currSum < 0)
        currSum = 0
    if (maxVal < currSum)
        maxVal = currSum;
}
```

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

O/P Normal Arr

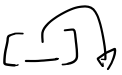
⇒ (5) ✓

O/P Circular Arr

⇒ (5, -2, 2, 1)

⇒ 6 ✓

eg [-1, -2]

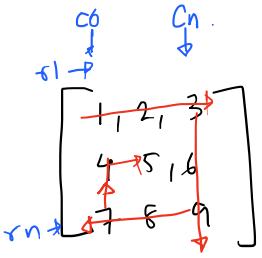


i	currSum	maxVal
0 (-1)	~~0~~ → ~~-1~~ → 0	~~-1~~ → 0 ✓
1 (-2)	0 → ~~-2~~ → 0	0 ✓

HW

Q

nxn



(1) CW

O/P 1, 2, 3, 6, 9, 8, 7, 4, 5