Arrays - Interview Problems

AGENDA:

- Max subarray sum of size k
- Sliding window
- Max consecutive 1's

Q1 Given array of N elements, print start and end index of all subarrays of size k (N >= k).

	0	1	2	3	4	5	6	7	8	9	10	11
arr[12] =	3	4	2	-1	6	7	8	9	3	2	-1	4

$k=3$

s	e
0	2
1	3
2	4
3	5
4	6
5	7
6	8
7	9
8	10
9	11

$k=6$

s	e
0	5
1	6
2	7
3	8
4	9
5	10
6	11

In general N elements, k size

s	e
0	k-1
1	k
2	k+1
3	k+2
⋮	⋮
N-k	N-1

Quiz 1

```
solve ( N, k ) {
    s = 0
    e = k-1
    while(e<N) {
        print(s,e)
        s += 1
        e += 1
    }
}
```

Total iterations = $N-k+1$

Time = $O(N)$

Space = $O(1)$

Quiz 2

Given N array elements, how many subarrays of len = k = $N-k+1$

Q2 Given array of N elements, find max subarray sum of length=k (N >= k).

	0	1	2	3	4	5	6	7	8	9
ar[10] =	-3	4	-2	5	3	-2	8	2	-1	4

$k=5$

s	e	sum
0	4	7
1	5	8
2	6	12
3	7	16
4	8	10
5	9	11

Ans = 16

Brute force Idea

→ For every subarray, get the sum & take their maximum.

```
ans = -∞
s=0, e=k-1
while (e<n) {
    // subarray [s e]
    sum=0
    for (i=s; i<=e; i++)
        sum += A[i]
    ans= max (ans, sum)
    s++, e++
}
return ans
```

Outer Loop = N-k+1

Inner Loop = k

If $k=1 \Rightarrow N \times 1 = N$

If $k=N \Rightarrow 1 \times N = N$

If $k=\frac{N}{2} \Rightarrow \left(\frac{N}{2}+1\right) \times \frac{N}{2} =$

TC : $O(N^2)$

Optimisation 1

sum [s e] → Prefix Sum

1) Construct pf [N]

2) For every subarray, use pf [ ] to get the sum.

TODO

TC : O(N)

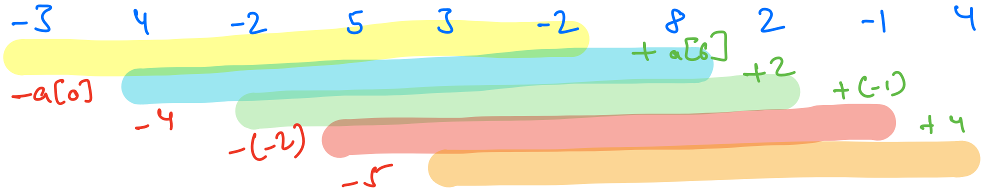
SC : O(N) ← Extra space for pf [N]

Solve without using any extra space or modifying the given array.

TC: O(n)

	0	1	2	3	4	5	6	7	8	9
ar[10] =	-3	4	-2	5	3	-2	8	2	-1	4

k=6



s	e	sum	
0	5	5	
1	6	5	− ar[0] + a[6] = 5 - (-3) + 8 = 16
2	7	16	− ar[1] + ar[7] = 14
3	8	14	− ar[2] + ar[8] = 15
4	9	15	− ar[3] + ar[9] = 14

Carry forward + Every window is of same size = sliding window

$a[N] = a_0 \quad a_1 \quad a_2 \quad .... \quad a_{k-1} \quad a_k \quad a_{k+1} \quad ...... \quad a_{n-1}$

$-a_0 \qquad +a_k$

$-a_1 \qquad +a_{k+1}$

s	e	sum
		Iterate & get the sum
0	k-1	
1	k	$sum = sum - a_0 + a_k$
2	k+1	$sum = sum - a_1 + a_{k+1}$
⋮	⋮	
s	e	$sum = sum - a_{s-1} + a_e$

Pseudocode

```
solve( int []A, int n, int k) {
    // Sum from 0 to k-1
    sum = 0
    for( i=0; i<k ; i++)                 } K
        sum += A[i]

    s=1, e=k

    ans = sum
    while ( e<n ) {
        // subarray [s  e] <- sum
        sum = sum - a[s-1] + a[e]
        s++, e++
        ans = max (ans, sum)
    }                                         } N-k
    return ans
}
```

Total iterations = $K + N - K$

Quiz 3 = N

Space = $O(1)$

Q3. Max consecutive 1's

0, 1

Given a binary arr[], we can atmost replace a single 0 with 1.

Find the max consecutive 1's we can get in the arr[].

Example 1

A = 0 1 1 1 0 1 1 0 1 1 0

× ×

0+ 3+1 = 4

3+2+1 = 6

2+2+1=5

2+0+1=3

Example 2

A = 1 1 1 1 1 1

ans = 6

if all ones

ans = N

Example 3

Quiz 4

A = 1 1 1 0 1 1 0 1 1 1 1 0 0 1 1 0 1 1

× ×

4+0+1 =5   2+ 2+1 = 5

3 + 2+1 = 6   2+4+1 = 7

0+ 2+1=3

Ex4

0 0 0

Idea & Pseudocode

```
Iterate on array
    if arr[i] ==1  → Do nothing
    if arr[i] ==0 :
        we can replace this 0 with 1

        Get the count of 1s on left = L
        Get the count of 1s on right = R
        Total ones = L + R + 1

        ans = max (ans, ones)
```

```
maxConsecutiveOnesReplace(int arr[]) {
        n = arr.length
        c = 0
        for ( i=0; i<n; i++) {
              if ( arr[i] == 1)
                    c = c+1                      Special
        }                                        Case
        if ( c == N)
              return N
        ans = 1
        for (i=0; i<n; i++)  {
              if ( arr[i] ==1)
                    continue

              // Current element is 0

              L=0
              for( j= i-1 ; j>=0 ; j--) {
                    if( arr[i] ==1)
                          L ++
                    else
                          break
              }
              R =0
              for (j= i+1; j<n ; j++) {
                    if ( arr[i] ==1)
                          R ++
                    else
                          break
              }
              ones  =  L+R+1
              ans  = max (ans , ones)
        }
        return ans
}
```

Quiz 5

Time Complexity ?

Time Complexity = $O(N)$

How is time complexity O(N) ?

How many iterations are there in this code?

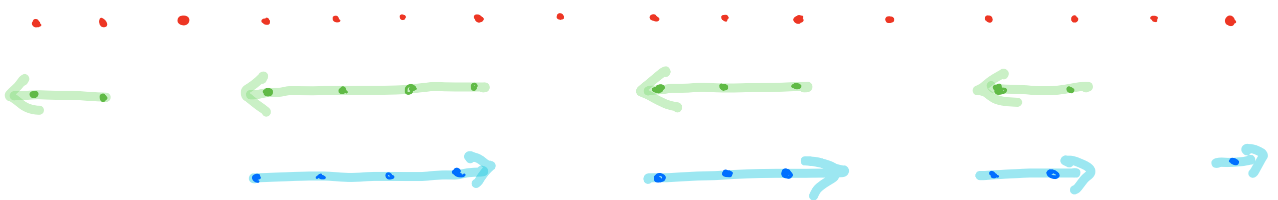
```
for ( l=1; l<=4; l++) {
    for ( i=0; i<N; i++) {

    }
}
```

Total Iterations = 4N

Time = O(N)

Every element will be visited 4 times

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	1	1	1	0	1	1	1	0	1	1	0	1



2 + 4 + 1 = 7 4+3+1 = 8 3+2+1 = 6 2+1+1 = 4

What is the max no of times any element is visited = 3 times

Total iterations in worst case = 3 N

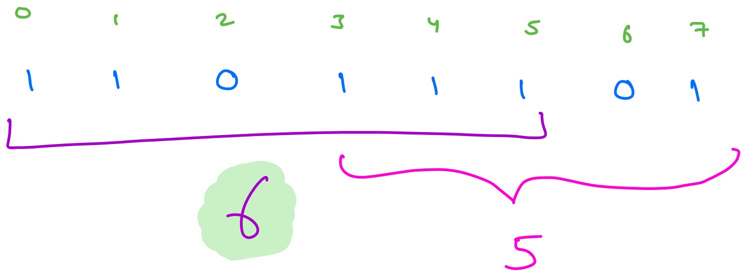
Time = $O(N)$

Q4

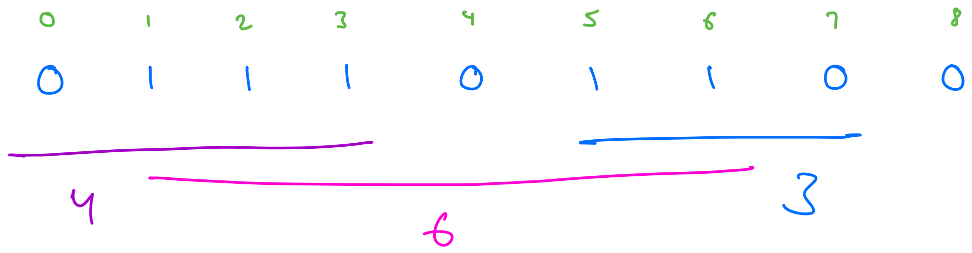
Given a binary arr[], we can atmost swap a single 0 with 1.

Find the max consecutive 1's we can get in the arr[].

Example 1



Example 2



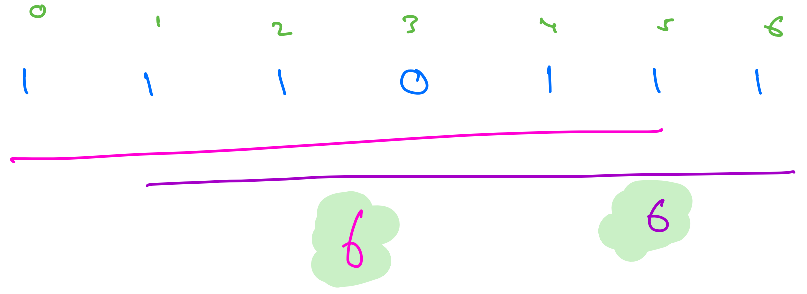
0	1	2	3	4	5	6	7	8
0	1	1	1	0	1	1	0	0

Count = 6

L+R = 6

Example 3

Quiz 6

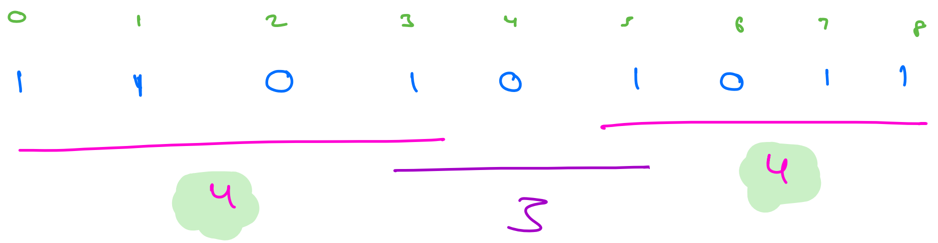


0	1	2	3	4	5	6
1	1	1	0	1	1	1

If we cannot spare an extra one =

Example 4

Quiz 7



0	1	2	3	4	5	6	7	8
1	1	0	1	0	1	0	1	1

Edge Case

if(L+R == total 1's)

— Ans = L+R

Avg Case ← else

— Same as previous problem

L+R+1

Ex 5

Count = 5
L+R = 5

0 1 1 1 0 1 1 0

If you have no spare ones

Ans =

To put 1 at index = i

you have to put a 0

on left side

↳ $(L-1) + R + 1$

or right side

↳ $L + (R-1) + 1$

$$\boxed{L + R}$$

```
maxConsecutiveOnesSwap(int arr[]) {
        n= arr.length
        c=0
        for( i=0; i<n; i++) {
              if ( arr[i] ==1)
                    c = c+1                          Special
        }                                            Case
        if ( c== N)
              return N
        ans = 0
        for (i=0; i<n; i++) {
              if (arr[i] ==1)
                    continue

              // Current element is 0

              L=0
              for( j= i-1 ; j>=0 ; j--) {
                    if( arr[i] ==1)
                          L++
                    else
                          break
              }
              R=0
              for (j=i+1; j<n; j++) {
                    if ( arr[i] ==1)
                          R++
                    else
                          break
              }
              if (L+R ==c)
                    ones = L+R
              else
                    ones = L+R+1
              ans = max (ans, ones)
        }
        return ans
}
```

Time = O(N)

Space = O(1)

Doubts

Thank
You

Prefix Array

Left to Right

Suffix Array

Right to Left

A	=	5	9	2	
pre	=	5	14	16	→
suff	=	16	11	2	←

```
suf [N]
suf [N-1] = A [N-1]

for ( i=n-2 ; i>=0; i--)
    suf [i] = suf [i+1] + A[i]
```

Prefix Sum - HW

$$B[i] = \frac{\text{product of all array elements}}{Arr[i]}$$

Do not use division operator

Good Night

Thank You

Friday