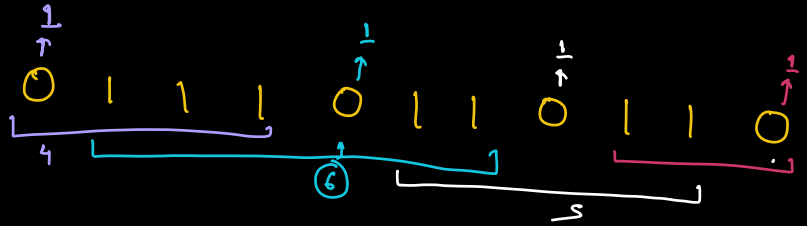
MS
LinkedIn
Amazon

Q Given a Binary array → all elements are 0/1

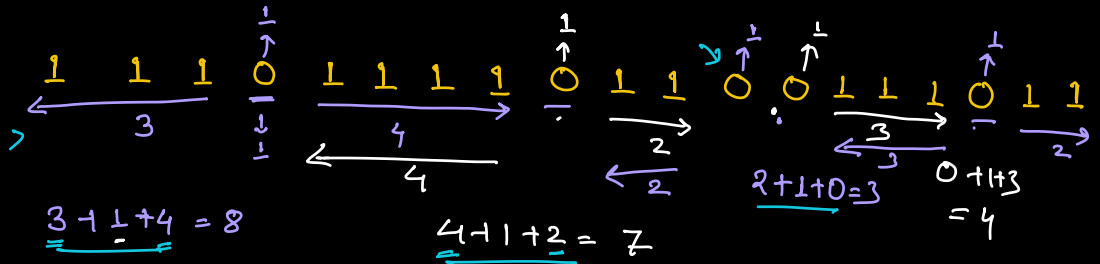
We are allowed to replace almost one 0 with a 1.

Return the length of max consecutive 1s.

Ex 1



Ex 2



Obs

For every 0, length of cons. 1s after replacing that 0

⇒ No. cons 1s on left side + 1 + No. cons. 1s on right side

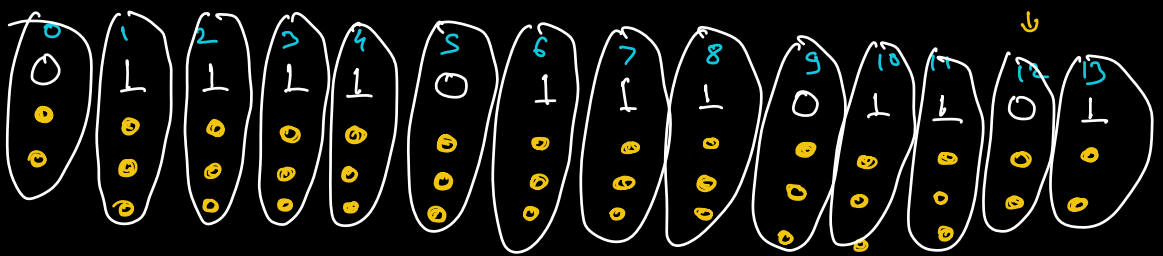
A : 1 1 1 1 ⇒ 4

N

```
ans = 0;
for (i=0; i<N; i++)
    if (arr[i] == 0) {
        // l = no. of cons. 1s on left;
        l = 0;
        for (j=i-1; j>=0; j--) {
            if (arr[j] == 1) l++;
            else { break; }

        // r = no. of cons 1s on right;
        r = 0;
        for (j=i+1; j<N; j++) {
            if (arr[j] == 1) { r++; }
            else {break;}

        ans = max (ans, l+r+1);
    }
if (ans == 0) ans = N;
```



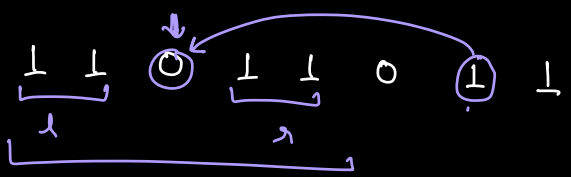
# of iterations ≈ 3N

TC : O(N)

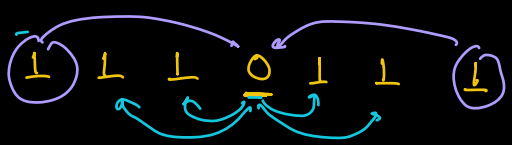
Given a Binary array → all elements are 0/1

We are allowed to swap almost one 0 with a 1.

Return the length of max consecutive 1s.

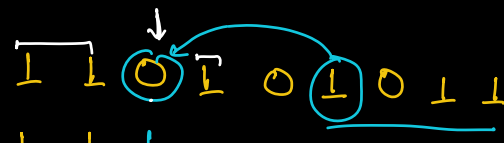


Quiz



=> 0 1 1 1 1 1 1

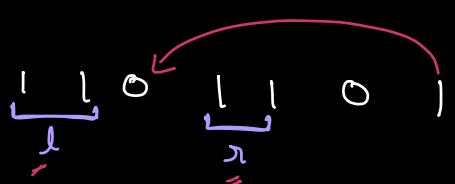
Quiz



=> Total 1s = 6

=> l+r = 3

l+r+1



Total 1s = 5

l+r = 4

if (Total > l+r)

(l+r)+1

Total 1s = 6

l+r = 6

=> l + r

if (total == l+r)

```
// iterate over array & count no. of 1s => Co
ans = 0;
for (i=0; i<N; i++)
    if (arr[i] == 0) {
        // l = no. of cons. 1s on left;
        l = 0;
        for (j=i-1; j>=0; j--) {
            if (arr[j] == 1) l++;
            else { break; }

        // r = no. of cons 1s on right;
        r = 0;
        for (j=i+1; j<N; j++) {
            if (arr[j] == 1) { r++; }
            else {break;}

        if (Co > l+r)
            len = l+r+1;
        else
            len = l+r;
    }
    ans = max(ans, len);
if (ans == 0) ret N;
```

TC : O(N)
SC : O(N)

Google

Find no. of triplets

i, j, k [indices]

such that

$i < j < k$

& $a[i] < a[j] < a[k]$

Ex

0	1	2	3	4
3	4	6	9	2

i	j	k	a[i]	a[j]	a[k]	
0	1	2	3	< 4	< 6	✓
0	1	3	3	< 4	< 9	✓
0	1	4	3	< 4	> 2	✗
1	2	3	4	< 6	< 9	✓
0	2	3	3	< 6	< 9	✓

⇒ 4

Quiz

$\overset{0}{2}, \overset{1}{6}, \overset{2}{9}, \overset{3}{4}, \overset{4}{10}$

i	j	K	a[i]	a[j]	a[k]
0	1	2	2	6	9 ✓
0	1	4	2	6	10 ✓
0	3	4	2	4	10 ✓
0	2	4	2	9	10 ✓
1	2	4	6	9	10 ✓

```
count = 0;
for (i=0; i<N; i++){
    for (j= i+1; j<N; j++) {
        if (a[i] > a[j])
            for (k = j+1; k<N ; k++){
                if ( a[i] < a[j] < a[k])
                    count++;
            }
    }
}
```

TC : $O(N^3)$

SC : $O(1)$

100 — 3, 6, 9, 12, 16, 11, 1

(0, 2, 3) (1, 2, 3)
(0, 2, 4) (1, 2, 4)
(0, 2, 5) (1, 2, 5)

if j = 2 a[j] ⇒ 9

i ⇒ 0, 1 ⇒ {3, 6} < 9

j ⇒ 3, 4, 5 ⇒ {12, 16, 11} > 9

3, 3, 100, 3, 9, 10, 10, 10

(0, 4, 5) (1, 4, 5) (3, 4, 5)
(0, 4, 6) (1, 4, 6) (3, 4, 6)
(0, 4, 7) (1, 4, 7) (3, 4, 7)

if j = 4

i ⇒ 0, 1, 3

K ⇒ 5, 6, 7

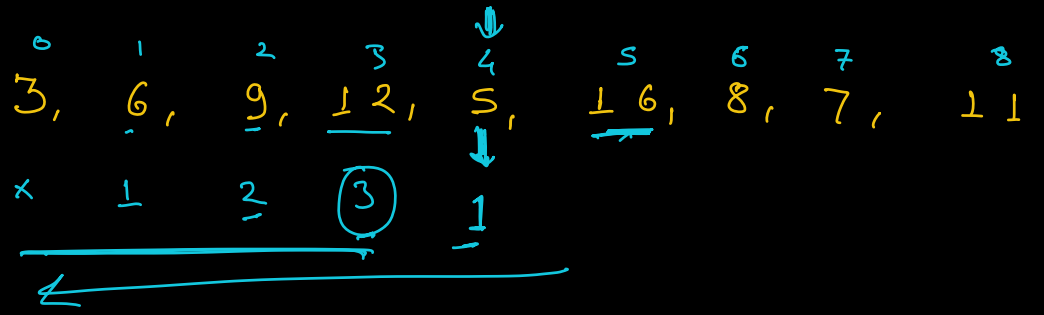
	0	1	2	3	4	5	6	7	8
	3,	6,	9,	12,	5,	16,	8,	7,	11
lefts		1	2	3	1	5	3	3	i
Rights		6	3	1	4	0	1	1	

6×1 3×2 3×1 4×1 5×0 3×1 3×1

$6 + 6 + 3 + 4 + 0 + 3 + 3 \Rightarrow 25$

```
Count = 0;
for (i = 1; i < N-1; i++) {
    left_smaller_count = No. of smaller elements on left of a[i]  [0, i-1]  => O(N)
    right_greater_count = No. of elements > a[i] on right of a[i]  [i+1, N-1]  O(N)
    Count = Count + (left_smaller_cnt * right_greater_cnt)
}
```

TC : $O(N^2)$



BST $\Rightarrow O(n \log n)^*$

Q Given an array of size N.
Print the start & end index of all the sub arrays of a given length K.

A: 3, 4, 2, -1, 6, 7, 8, 9, 3, 2, -1, 4 | K = 6

K = 1

[0, 0]
[1, 1]
[2, 2]
[N-1, N-1]
[11, 11]

12 [N]

K = 2

[0, 1]
[1, 2]
[2, 3]
[9, 10] [N-2, N-1]
[10, 11]

11 (N-1)

K = 3

[0, 2]
[1, 3]
[N-3, N-1]
[9, 11]

10 (N-2)

K = 6

[0, 5]
[1, 6]
[2, 7]
[3, 8]
[4, 9]
[5, 10] [N-6, N-1]
[6, 11]

7 (N-5)

K

[0, K-1]
[1, K]
[2, K+1]
[N-K, N-1]

N-K+1

```
s = 0;
e = k-1;
while (s <= (N-k)) {          // e < n
    Print (s, e);
    s++;
    e++;
}
```

TC: $O(N-K)$

Learning: How to iterate over all subarrays of size K

Q Facebook Amazon MS

Find the max subarray sum with length = K.

	0	1	2	3	4	5	6	7	8	9
A :	-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 = -∞;
S = 0;
e = K-1;
while (S <= (N-K)) {          ⟹ # iterations = N-K+1
    sum = 0;
    for (i=S; i <= e; i++){   ⟹ # iterations = K
        sum += a[i];
    }
    ans = max(ans, sum);
    S++;
    e++;
}
```

Range Sum from S to e
sum = PS[e] - PS[S-1]

$$\text{Total iterations} = \frac{N(N-N+1)}{K(N-K+1)}$$

K : [1, N]

- K = 1 → iteration = N
- K = N/2 → iterations = $\frac{N}{2}\left(N-\frac{N}{2}+1\right)$
- K = N → iteration = N

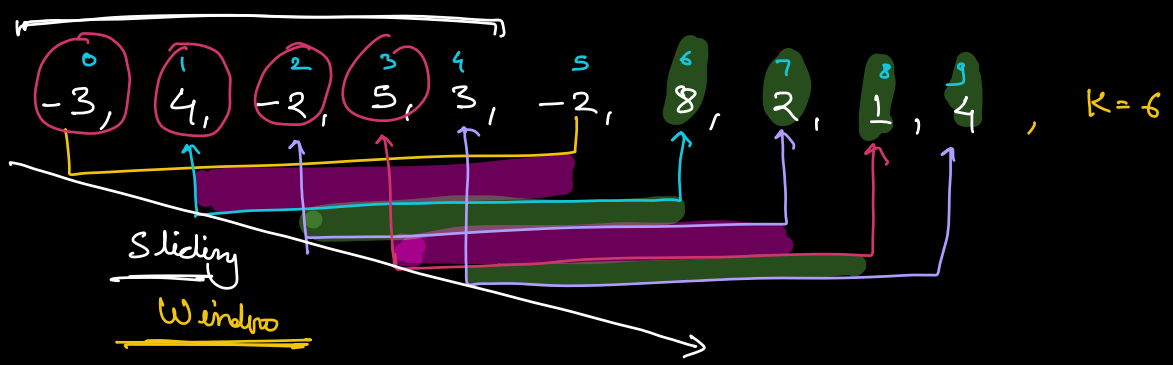
$$= \frac{N}{2} \times \left(\frac{N}{2}+1\right)$$

TC : $O(N^2)$

Use PS

TC : $O(N)$

SC : $O(N)$ {PS array}



$\Sigma[0,5]$: 5

$\Sigma[1,6]$: $\Sigma[0,5] - a[0] + a[6] \Rightarrow 5 - (-3) + 8 \rightarrow 16$

$\Sigma[2,7]$: $\Sigma[1,6] - a[1] + a[7] \Rightarrow 16 - 4 + 2 \rightarrow 14$

$\Sigma[3,8]$: $\Sigma[2,7] - a[2] + a[8] \Rightarrow 14 - (-2) + 1 \rightarrow 17$

$\Sigma[4,9]$: $\Sigma[3,8] - a[3] + a[9] \Rightarrow 17 - 5 + 4 \rightarrow 16$

;

$\Sigma[i,j]$: $\Sigma[i-1, j-1] - a[i-1] + a[j]$

// N sized array, K sized window.

⇒ 1st subary : $[0, K-1]$ ⇒ Iterate & get the sum → S

- 2nd subary : $[1, K] \Rightarrow S = S - a[0] + a[K]$
- 3rd subary : $[2, K+1] \Rightarrow S = S - a[1] + a[K+1]$
- ⋮
- last subary : $[N-K, N-1] \Rightarrow S = S - a[N-K-1] + a[N-1]$

```
Sum = 0;
for (i = 0; i < K; i++) {          // Sum of 1st K elements = Sum
    Sum += a[i]
}
```

iterations = K

```
s = 1;
e = K;
ans = Sum;
while (s <= (N-K)) {
    Sum = Sum - a[s-1] + a[e];
    ans = max(ans, sum);
    s++;
    e++;
}
```

(s-1) s —— e

iterations = [1, N-K] = N-K

Total iterations = K + (N-K)
= N

TC : O(N)

SC (Extra) : O(1)