



Max Consecutive Ones III

$arr[] = [1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0]$ $k = 2$





Max Consecutive Ones III

→ at most k 0's

arr[] = [1 1 1 0 0 0 1 1 1 1 0] k=2





Main Consecutive Dns 111

arr[] = [1 1 1 0 0 1 1 1 1 1] k=2

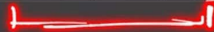
6





Max Consecutive Ones III → longest subarray with
max zeros as k.

arr[] = [1 1 1 0 0 0 1 1 1 1 0] k=2



6

2 zeros





Max Consecutive Ones III → longest subarray with
at most k zeros

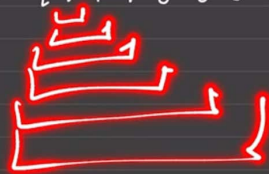
$arr[] = [1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0]$ $k=2$





Max Consecutive Ones III → longest subarray with
at most k zeros

arr[] = [1 1 1 0 0 0 1 1 1 1 0] $k=2$



~~zeros = 0 0 1 2 3~~





```
fun (i = 0 → n)
```

```
    zeros = 0
```

```
    fun (j = i → n)
```

```
        if (nums[j] == 0)
```

```
            zeros++;
```

```
        if (zeros <= k)
```

```
            len = j - i + 1
```



arr[] = [1 1 1 0 0 0 1 1 1 1 0] k=2

at most k zeros

maxLen = 0

for (i = 0 → n)

zeros = 0
for (j = i → n)

if (arr[j] == 0)
zeros++;

if (zeros ≤ k)

len = j - i + 1



arr[] = [1 1 1 0 0 0 1 1 1 1 0] k=2

maxLen = 0

for (i = 0 → n)

TC → $O(n^2)$
SC → $O(1)$

zeros = 0
for (j = i → n)

if (arr[j] == 0)
 zeros++;

if (zeros <= k)

 len = j - i + 1
 maxLen = max(maxLen, len);

?

else



arr[] = [1 1 1 0 0 0 1 1 1 1 0] k=2

maxLen = 0

for (i = 0 → n)

TC → $O(n^2)$
SC → $O(1)$

zeros = 0
for (j = i → n)

if (arr[j] == 0)
 zeros++;

if (zeros ≤ k)

 len = j - i + 1
 maxLen = max(maxLen, len);

?

else





Max Consecutive Ones III → longest subarray with at most k zeros

arr[] = [1 1 1 0 0 0 1 1 1 1 0] $k=2$
 l r

zeros = 0 1 2

max len = 0 1 2 3 4 5

len = $r - l + 1$





Max Consecutive Ones III → longest subarray with at most k zeros

arr[] = [1 1 1 0 0 0 1 1 1 1 0] $k=2$

zeros = 0 1 2 3 4 5
2

maxLen = 0 1 2 3 4 5 6

len = r - l + 1



arr[] = [1 1 1 0 0 0 1 1 1 1 0] k=2
 \nwarrow
 \sim

func (list<int> nums, k)

man len = 0, l = 0, r = 0





~

```
func (list <int> nums, l <)
```

```
man len = 0, l = 0, r = 0, zeros = 0
```

```
while (r < nums.size())
```

```
{  
    if (nums[r] == 0) zeros++;
```





```
func (list <int> nums, k)
```

```
{  
    maxlen = 0, l = 0, r = 0, zeros = 0
```

```
while (r < nums.size())  
{
```

```
    if (nums[r] == 0) zeros++;
```

```
    while (zeros > k)
```

```
    {  
        if (nums[l] == 0) zeros--;  
        l++;
```

```
    }
```





manLen = 0, l = 0, r = -1, zeros = -1

```
while (r < nums.size())  
{
```

```
    if (nums[r] == 0) zeros++;
```

```
    while (zeros > k)  
    {
```

```
        if (nums[l] == 0) zeros--;  
        l++;
```

```
    }
```

```
    if (zeros <= k) { len = r - l + 1;  
                    manLen = max(len, manLen);  
    }
```





```
U
while (zeros > k)
{
    if (a[left] == 0) zeros--;
    left++;
}
if (zeros <= k) { len = r - l + 1;
                  maxlen = max(len, maxlen);
                  }

right++;
}
```





1 1 1 1 1 1 0 0
x x x x x x x ↑
x
—
L

$k = 1$

~~Zeros = 2~~ ||





Max Consecutive Ones III → longest subarray with at most k zeros

arr[] = [1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0] k=2

(Note: In the original image, the first three 1s are circled in red, the first 0 is boxed in yellow, and the last 1 is circled in red. There are also 'x' marks under the first four elements.)

func (list <int> nums, k)

TC → $O(n) + O(n)$
= $O(2n)$

maxlen = 0, l = 0, r = 0, zeros = 0

→ while (r < nums.size())

if (nums[r] == 0) zeros++;

→ while (zeros > k)

if (nums[l] == 0) zeros--;





```
func (list <int> nums, k)
```

```
    maxlen = 0, l = 0, r = 0, zeros = 0
```

```
    while (r < nums.size())
```

```
        if (nums[r] == 0) zeros++;
```

```
        while (zeros > k)
```

```
            if (nums[l] == 0) zeros--;
```

```
            l++;
```

```
        }
```

```
        if (zeros <= k) { len = r - l + 1;
```

```
            maxlen = max(len, maxlen);
```

TC $\rightarrow O(N) + O(N)$

$= O(2N)$

SC $\rightarrow O(1)$





Max Consecutive Ones III → longest subarray with
at most k zeros

arr[] = [1 1 1 0 0 0 1 1 1 1 0] $k=2$
 l r

zeros = 0 1 2 3

max len = 4 5 6 7 8





Max Consecutive Ones III → longest subarray with
at most k zeros

arr[] = [1, 1, 1, 0, 0, 0, 1, 1, 1, 0] $k=2$

zeros = 0, 1, 2, 3

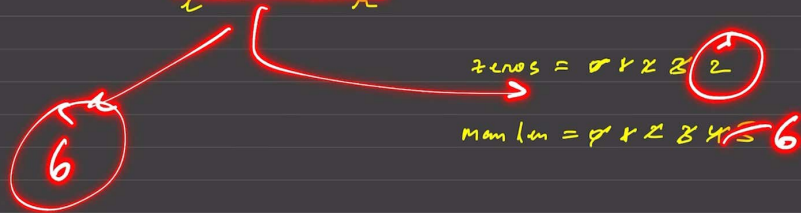
max len = 4, 5, 6, 7, 8





Max Consecutive Ones III → longest subarray with
at most k zeros

arr[] = [1 1 1 0 0 0 1 1 1 1 0] $k=2$





can consecutively

at most k zeros

arr[] = [1 1 1 0 0 0 1 1 1 1 0] $k=2$



Main Consecutive Dns 111 → longest subarray with
at most k zeros

arr[] = [1 1 1 0 0 0 1 1 1 1 0] k=2
 L ~

zeros = 0 1 2 3 4 5 6

max len = 0 1 2 3 4 5 6





```
func (list <int> nums, k)
```

```
{
```

```
    l=0, r=0, zeros=0, maxlen=0    zeros += |nums|
```

```
    while (r < nums.size)
```

```
    {
```

```
        if (nums[r] == 0) zeros++
```





```
while (n < nums.size)
```

```
{
```

```
    if (nums[n] == 0) zeros ++
```

```
    if (zeros > 1)
```

```
    {
```

```
        if (nums[left] == 0) zeros --;  
        left ++;
```

```
    }
```

```
    if (zeros <= 1) len = n - left + 1, maxlen =  
        max(maxlen,
```





```
if (zeros > 1)
{
    if (nums[left] == 0) zeros--;
    left++;
}
if (zeros <= 1) len = r - l + 1, maxlen =
max(maxlen, len)
r++;
}
```





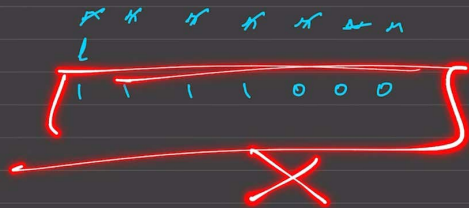
```
if (zeros > 1)
{
    if (nums[left] == 0) zeros--;
    left++;
}
if (zeros <= 1) len = n - l + 1, maxlen =
max(maxlen, len)
n++;
}
return maxlen;
}
```





```
if (zeros > 1)
{
    if (nums[left] == 0) zeros--;
    left++;
}
if (zeros <= 1) len = n - l + 1, maxlen =
max(maxlen, len)
n++;
}
return maxlen;
}
```





$$k = 2$$

$$\text{zeros} = n - 3$$





```
func (list <int> nums, k)
```

TC $\Rightarrow O(N)$

```
{  
    l=0, r=0, zeros=0, maxlen=0
```

```
while (r < nums.size)  
{
```

```
    if (nums[r] == 0) zeros++
```

```
    → if (zeros > k)  
    {
```

```
        if (nums[left] == 0) zeros--;  
        left++;  
    }
```

```
    if (zeros <= k) len = r - l + 1, maxlen =  
        max(maxlen, len)
```

```
    r++;
```

```
}
```

```
return maxlen;
```



Main Consecutive Dns III → longest subarray with
at most k zeros

arr[] = [1 1 1 0 0 0 1 1 1 1 0] $k=2$

```
func (list<int> nums, k)
```

TC → $O(N)$

```
{  
    l=0, r=0, zeros=0, maxlen=0
```

```
    while (r < nums.size())  
    {
```

```
        if (nums[r] == 0) zeros++
```

```
        → if (zeros > k)  
        {
```

```
            if (nums[l] == 0) zeros--;  
            l++;  
        }
```





$$k = 2$$

$$\text{zeros} = x \approx 3$$

$$\text{min len} = 6$$

