Max Consecutive Ones III

arr[] = [1 1 1 0 0 0 1 1 1 1 0] K = 2

Max Consecutive Ones III

→ atmost k 0s

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

Max Consecutive Ones III

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

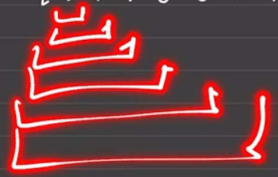
2 zeros

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

```
for(i = 0 → n)
{
    zeros = 0
    for(j = i → n)
    {
        if (nums[j] == 0)
            zeros++;

        if (zeros <= k)
        {
            len = j - i + 1
```

at most K zeros

arr[] = [1 1 1 0 0 0 1 1 1 1 0] K = 2

i j

```
maxLen = 0

for(i = 0 → n)
{
    zeros = 0
    for(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

maxlen = 0

for (i = 0 → n)
{
  zeros = 0
  for (j = i → n)
  {
    if (nums[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)
{
  zeros = 0
  for (j = i → n)
  {
    if (nums[j] == 0)
      zeros++;

    if (zeros <= k)
    {
      len = j - i + 1
      maxlen = max(maxlen, len);
    }
    else
```

TC → O(n²)
SC → 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

maxLen = 0 1 2 3 4 5

len = r - l + 1

TUF

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

maxLen = 0 1 2 3 4 5 6

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

l r

zeros = 0 1 2 3 4 5
2

maxLen = 0 1 2 3 4 5 6

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
        l   r

zeros = 0 1 2 3 3 3
            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

l
r

fun (list<int> nums , k)

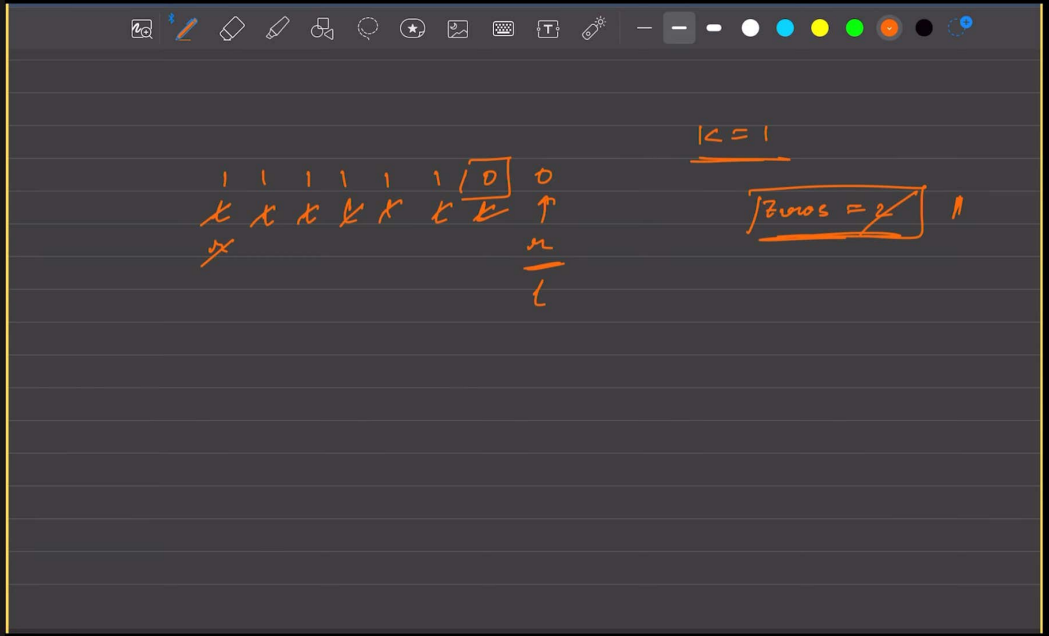
maxLen = 0 , l = 0 , r = 0

```
fun ( list<int> nums , k )
{
    maxLen = 0 , l = 0 , r = 0 , zeros = 0

    while ( r < nums.size() )
    {
        if ( nums[r] == 0 ) zeros++;
```

```
fun ( 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 ( a[left] == 0 ) zeros--;
                left++;
        }
```

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

while (r < nums.size())
{
    if (nums[r] == 0) zeros++;

    while (zeros > k)
    {
        if (a[left] == 0) zeros--;
        left++;
    }

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

```
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
n
1
k = 1
Bulbs = 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

l r r r r r r

TC → O(N) + O(N) = O(2N)

```
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 ( a[l] == 0) zeros--;
```

```
k k k k        n

TC → O(N) + O(N)
   = O(2N)
SC → O(1)

fun ( 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 ( a[left] == 0 ) zeros--;
            left++;
        }

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


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

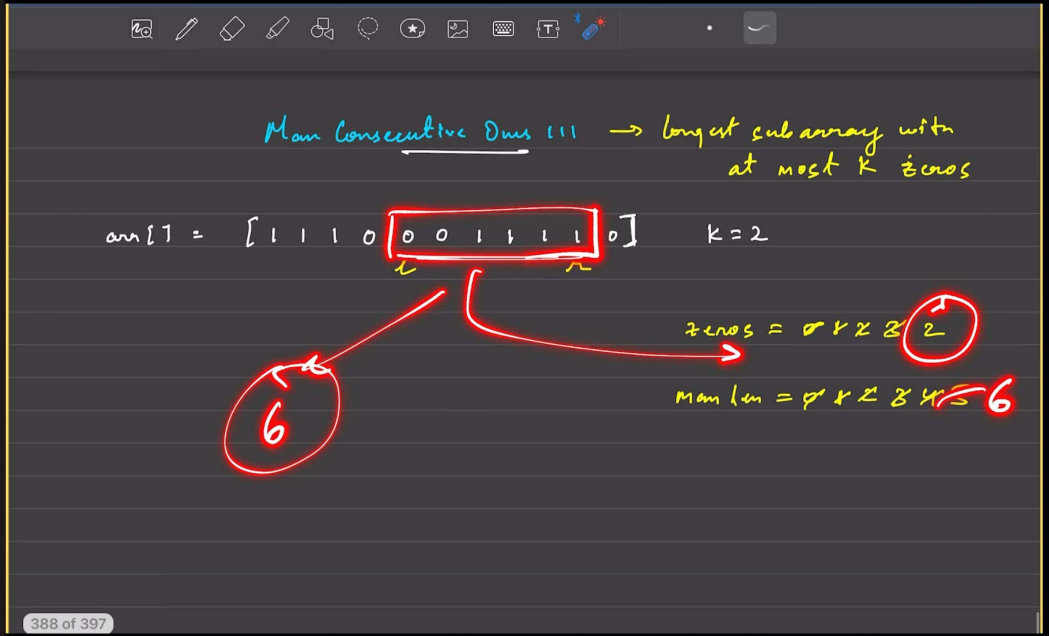
max len = 0 1 2 3 4 5

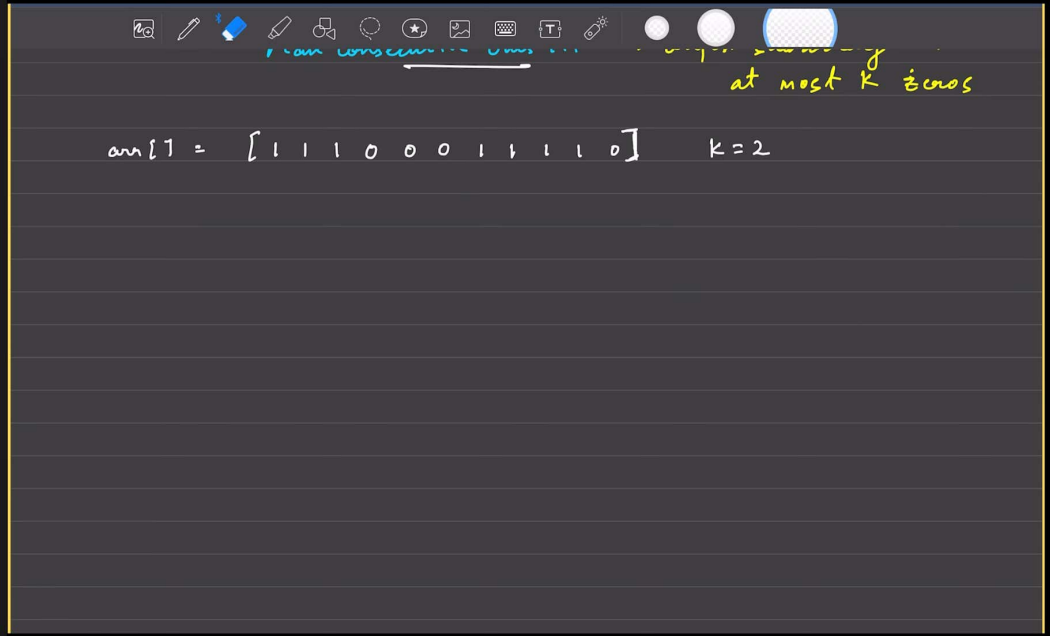
Max Consecutive Ones III → longest subarray with at most K zeros

arr[] = [1 1 1 0 0 0 1 1 1 1 0] K = 2

zeros = 0 1 2 3

max len = 0 1 2 3 4 5


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
2
max len = 1 2 3 4 5 6
6
388 of 397


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

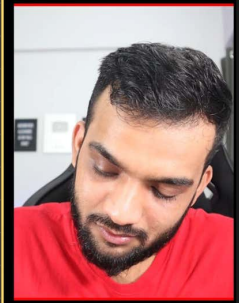
arr[] = [1 1 1 0 0 0 1 1 1 1 0] k = 2

L R

zeros = 0 1 2 3

max len = 6

```
func ( list<int> nums , k)
{
    l=0 , r=0 , zeros=0 , maxlen=0        zeros += !nums[r]

    while (r < nums.size)
    {
        if (nums[r] == 0) zeros++
```

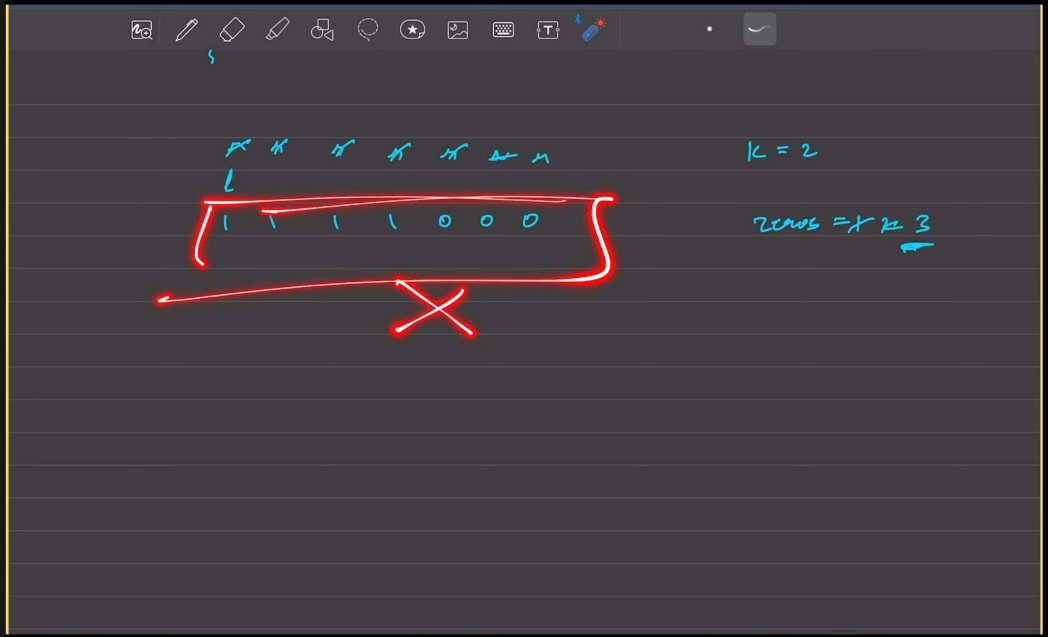


```
while (r < nums.size)
{
    if (nums[r] == 0) zeros++

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

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

r++;
}
```

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

        r++;
    }
    return maxLen;

}
```

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

        r++;
    }
    return maxLen;
}
```


1 1 1 1 0 0 0
k = 2
zeros = 1 x 3

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

        r++ ;
    }
```

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

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

$k = 2$

zeros = $k \times 3$

min len = 6