**Problem Statement: Binary Array and Magic Stick**

Cheems has been given a binary array of size N. He wants to maximize the length of the longest consecutive 1's. Cheems has a magic stick whose power is K. With this magic stick, he can flip 0's to 1's at most K times.

You need to find the length of the longest consecutive 1's that can be achieved with the help of this magic stick.

**Input Format**

* The first line contains two space-separated integers N and K.
* The second line contains N integers describing the array.

**Constraints**

* 1 <= N <= 10000

**Output Format**

* An integer representing the length of the longest consecutive 1's.

**Sample Input**

11 2

1 1 1 0 0 0 1 1 1 1 0

**Sample Output**

6

**Explanation**

The longest subarray length of consecutive 1's is 6. The subarray from index 5 to 10 can be transformed into consecutive 1's after performing magic and flipping both zeroes.

**Test Cases**

**Test Case 1**

**Input:**

11 2

1 1 1 0 0 0 1 1 1 1 0

**Output:**

6

**Explanation:** The longest subarray of consecutive 1's is [5,10], which can be achieved by flipping the two 0's at index 4 and 5.

**Test Case 2**

**Input:**

10 3

0 0 0 1 1 0 1 0 1 1

**Output:**

8

**Explanation:** By flipping three 0's at index 0, 1, and 2, we get the longest subarray of consecutive 1's as [0,6].

**Test Case 3**

**Input:**

6 1

1 0 1 0 1 1

**Output:**

4

**Explanation:** Flipping the 0 at index 1 or index 3 results in a maximum subarray length of 4.

**Test Case 4**

**Input:**

5 0

1 0 1 0 1

**Output:**

1

**Explanation:** No flips are allowed, so the longest subarray of consecutive 1's is just 1.

**Test Case 5**

**Input:**

7 5

0 0 0 0 0 0 0

**Output:**

5

**Explanation:** The longest subarray of consecutive 1's is 5 after flipping any five 0's.

**Solution**

To solve this problem, we can use a sliding window (or two-pointer) approach. The idea is to keep a window that contains at most KKK zeros and track the maximum length of the window containing consecutive 1's.

Here is the Python code to solve the problem:

python

def longest\_consecutive\_ones(arr, n, k):

left = 0

max\_length = 0

zero\_count = 0

for right in range(n):

if arr[right] == 0:

zero\_count += 1

while zero\_count > k:

if arr[left] == 0:

zero\_count -= 1

left += 1

max\_length = max(max\_length, right - left + 1)

return max\_length

# Sample Input 1

n, k = 11, 2

arr = [1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0]

print(longest\_consecutive\_ones(arr, n, k)) # Output: 6

# Sample Input 2

n, k = 10, 3

arr = [0, 0, 0, 1, 1, 0, 1, 0, 1, 1]

print(longest\_consecutive\_ones(arr, n, k)) # Output: 7

# Sample Input 3

n, k = 6, 1

arr = [1, 0, 1, 0, 1, 1]

print(longest\_consecutive\_ones(arr, n, k)) # Output: 4

# Sample Input 4

n, k = 5, 0

arr = [1, 0, 1, 0, 1]

print(longest\_consecutive\_ones(arr, n, k)) # Output: 1

# Sample Input 5

n, k = 7, 5

arr = [0, 0, 0, 0, 0, 0, 0]

print(longest\_consecutive\_ones(arr, n, k)) # Output: 5

This approach ensures we efficiently find the longest subarray of 1's with at most kkk flips using a sliding window technique