function length\_of\_longest\_substring(arr, k) {

let windowStart = 0,

maxLength = 0,

maxOnesCount = 0;

// Try to extend the range [windowStart, windowEnd]

for (windowEnd = 0; windowEnd < arr.length; windowEnd++) {

if (arr[windowEnd] === 1) {

maxOnesCount += 1;

}

// Current window size is from windowStart to windowEnd, overall we have a maximum of 1s

// repeating 'maxOnesCount' times, this means we can have a window with 'maxOnesCount' 1s

// and the remaining are 0s which should replace with 1s.

// now, if the remaining 1s are more than 'k', it is the time to shrink the window as we

// are not allowed to replace more than 'k' 0s

if ((windowEnd - windowStart + 1 - maxOnesCount) > k) {

if (arr[windowStart] === 1) {

maxOnesCount -= 1;

}

windowStart += 1;

}

maxLength = Math.max(maxLength, windowEnd - windowStart + 1);

}

return maxLength;

}

console.log(length\_of\_longest\_substring([0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1], 2));

console.log(length\_of\_longest\_substring([0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1], 3));

C++

using namespace std;

#include <iostream>

#include <vector>

class ReplacingOnes {

public:

static int findLength(const vector<int>& arr, int k) {

int windowStart = 0, maxLength = 0, maxOnesCount = 0;

// try to extend the range [windowStart, windowEnd]

for (int windowEnd = 0; windowEnd < arr.size(); windowEnd++) {

if (arr[windowEnd] == 1) {

maxOnesCount++;

}

// current window size is from windowStart to windowEnd, overall we have a maximum of 1s

// repeating a maximum of 'maxOnesCount' times, this means that we can have a window with

// 'maxOnesCount' 1s and the remaining are 0s which should replace with 1s.

// now, if the remaining 0s are more than 'k', it is the time to shrink the window as we

// are not allowed to replace more than 'k' Os

if (windowEnd - windowStart + 1 - maxOnesCount > k) {

if (arr[windowStart] == 1) {

maxOnesCount--;

}

windowStart++;

}

maxLength = max(maxLength, windowEnd - windowStart + 1);

}

return maxLength;

}

};

int main(int argc, char\* argv[]) {

cout << ReplacingOnes::findLength(vector<int>{0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1}, 2) << endl;

cout << ReplacingOnes::findLength(vector<int>{0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1}, 3) << endl;

}

Python

def length\_of\_longest\_substring(arr, k):

window\_start, max\_length, max\_ones\_count = 0, 0, 0

# Try to extend the range [window\_start, window\_end]

for window\_end in range(len(arr)):

if arr[window\_end] == 1:

max\_ones\_count += 1

# Current window size is from window\_start to window\_end, overall we have a maximum of 1s

# repeating 'max\_ones\_count' times, this means we can have a window with 'max\_ones\_count' 1s

# and the remaining are 0s which should replace with 1s.

# now, if the remaining 1s are more than 'k', it is the time to shrink the window as we

# are not allowed to replace more than 'k' 0s

if (window\_end - window\_start + 1 - max\_ones\_count) > k:

if arr[window\_start] == 1:

max\_ones\_count -= 1

window\_start += 1

max\_length = max(max\_length, window\_end - window\_start + 1)

return max\_length

def main():

print(length\_of\_longest\_substring([0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1], 2))

print(length\_of\_longest\_substring(

[0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1], 3))

main()

Java:

class ReplacingOnes {

public static int findLength(int[] arr, int k) {

int windowStart = 0, maxLength = 0, maxOnesCount = 0;

// try to extend the range [windowStart, windowEnd]

for (int windowEnd = 0; windowEnd < arr.length; windowEnd++) {

if (arr[windowEnd] == 1)

maxOnesCount++;

// current window size is from windowStart to windowEnd, overall we have a maximum of 1s

// repeating a maximum of 'maxOnesCount' times, this means that we can have a window with

// 'maxOnesCount' 1s and the remaining are 0s which should replace with 1s.

// now, if the remaining 0s are more than 'k', it is the time to shrink the window as we

// are not allowed to replace more than 'k' Os

if (windowEnd - windowStart + 1 - maxOnesCount > k) {

if (arr[windowStart] == 1)

maxOnesCount--;

windowStart++;

}

maxLength = Math.max(maxLength, windowEnd - windowStart + 1);

}

return maxLength;

}

public static void main(String[] args) {

System.out.println(ReplacingOnes.findLength(new int[] { 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1 }, 2));

System.out.println(ReplacingOnes.findLength(new int[] { 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1 }, 3));

}

}