|  |  |
| --- | --- |
| Problem Solving Workshop #2 | Tech Interviews and Competitive Programming Meetup |
| February 20, 2016 | <https://www.meetup.com/tech-interviews-and-competitive-programming>/ |

Instructor: Eugene Yarovoi (can be [contacted](https://www.meetup.com/tech-interviews-and-competitive-programming/members/100243892/) through the group Meetup page above under Organizers)

**More practice questions:** leetcode.com, glassdoor.com, geeksforgeeks.org

**Books:** Elements of Programming Interviews, Cracking the Coding Interview

**Have questions you want answered?** Contact the instructor, or ask on [Quora](https://www.quora.com/). You can post questions and [follow the instructor](https://www.quora.com/profile/Eugene-Yarovoi) and other people who write about algorithms.

Try to find optimized solutions, and provide a time and space complexity analysis with every solution for the algorithms questions.

**Easy Problem**

Given an array of 0s and 1s, and an integer k, find the longest contiguous streak of 1s that you can get by changing any k 0s to 1s.

**Example Input**: array = [1,1,0,0,1,1,1,0,1,1], k = 1

**Input Explanation**: We can change one (k=1) 0 to a 1 in the array

**Output**: 6

**Output Explanation**: if we change the 0 to a 1 at index 7 (counting from 0), we get a contiguous streak of 1s having length 6. This is the largest streak we can get. No other change gets us a bigger streak.

Medium Problem

Given an array and an integer k, find the maximum of every contiguous subarray of size k.

**Example Input**: array = [1, 2, 3, 1, 4, 5, 2, 3, 6], k = 3

**Explanation**: We want the maximum of each contiguous subarray of size 3, since k = 3. That means we first want to produce the max of {1, 2, 3}, then the max of {2, 3, 1}, and so on.

**Output**: return an array containing [3, 3, 4, 5, 5, 5, 6]

Hard Problem

Given an array of integers, find the contiguous subarray that produces the largest value when its elements are bitwise XOR’ed together. That is, if the array is *A*, let XorSubarray(i, j) = a[i] ^ a[i+1] ^ … ^ a[j-1] ^ a[j], where ^ is the bitwise exclusive or (XOR) operation. Find the i, j that maximize this function. Do this in better than O(n2) complexity.

**Example Input**: {1, 2, 3, 4}  
**Output**: [3, 4]  
**Explanation**: The subarray [3, 4] has maximum XOR value. 3 is 11 in binary, and 4 is 100 in binary. Their XOR is 111 = 7. No larger XOR can be achieved with any other subarray.