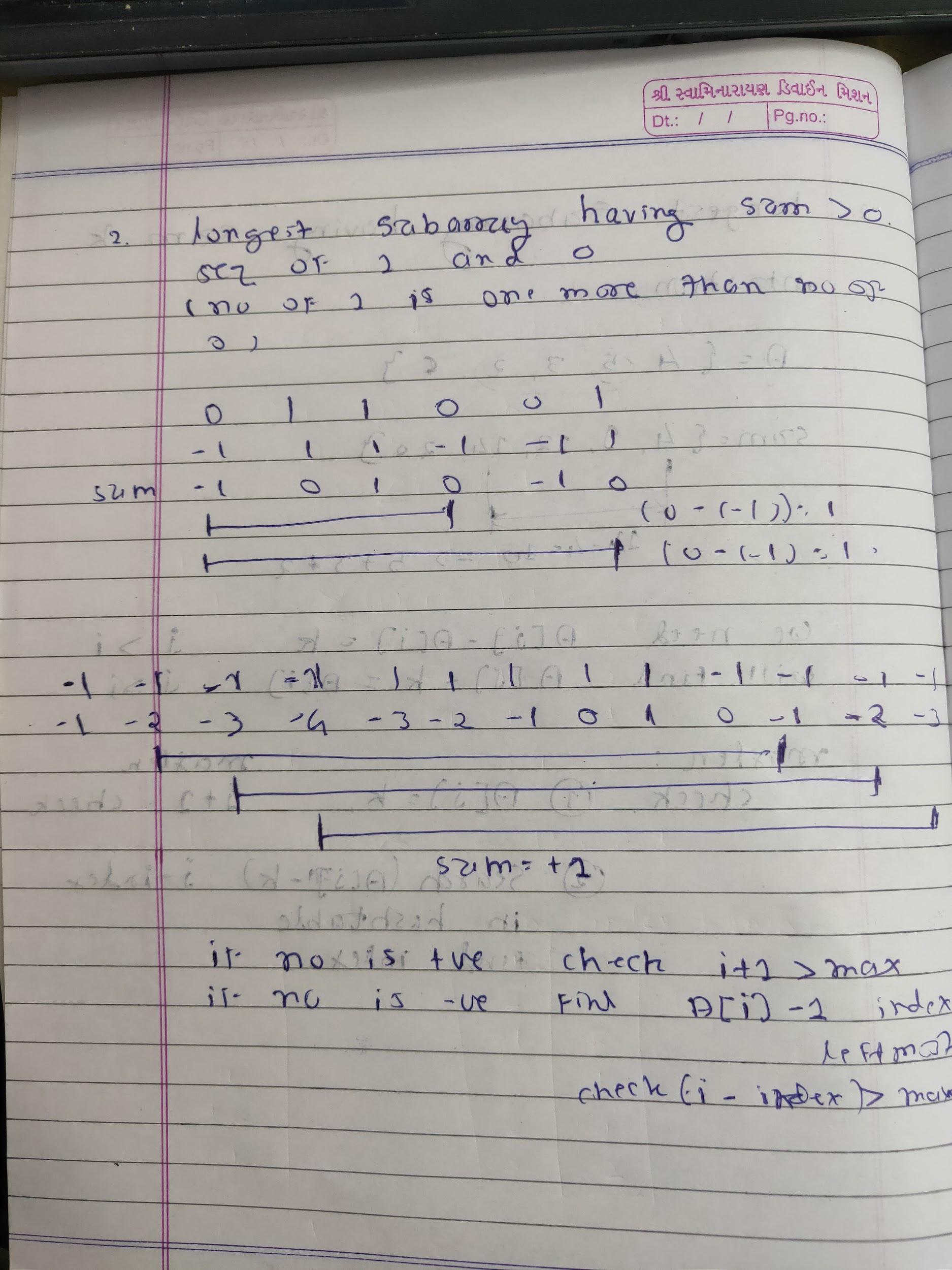
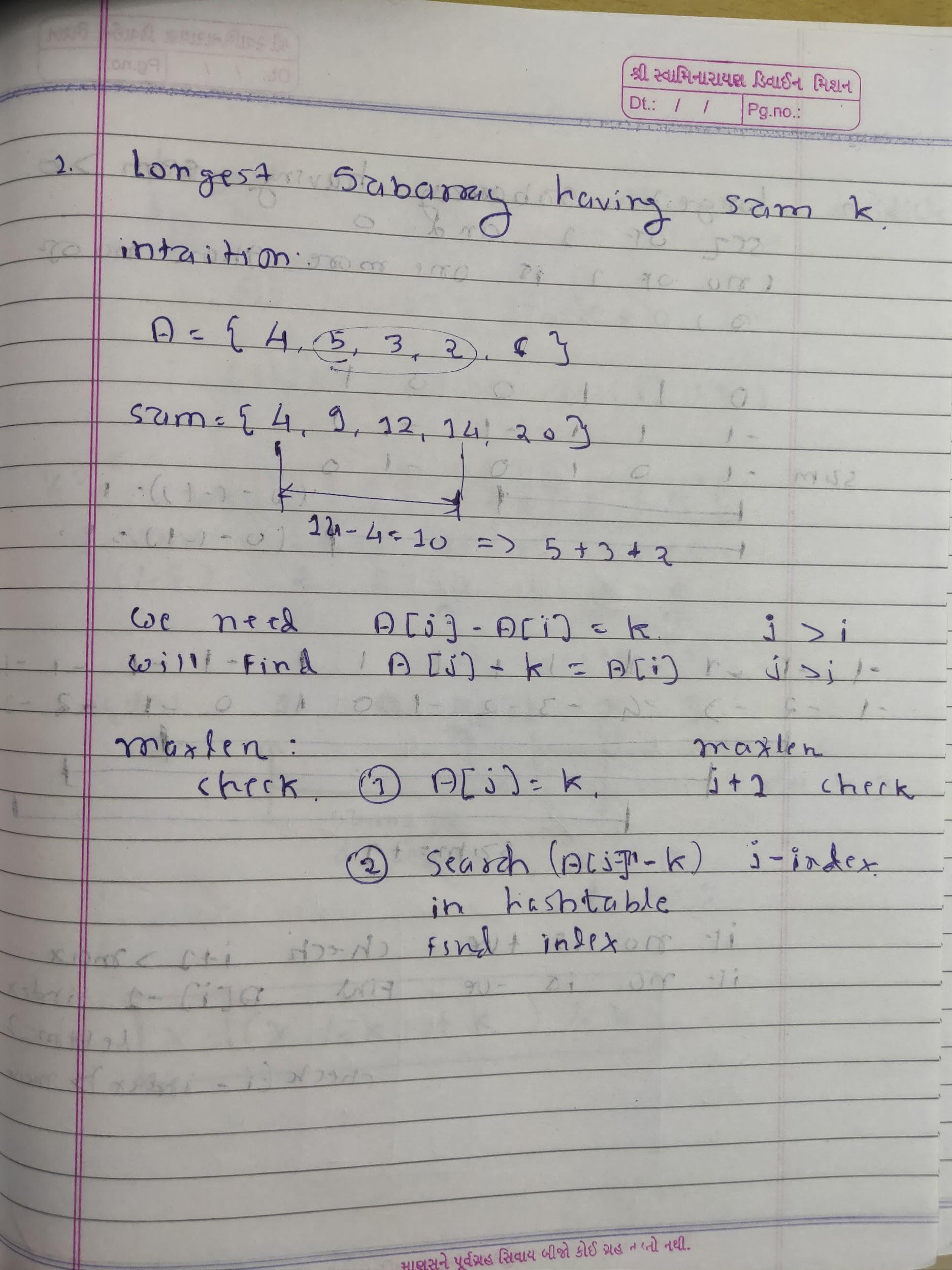
1. Longest subarray having sum > 0 in sequence of 0 and 1

<https://leetcode.com/problems/longest-well-performing-interval/submissions/>



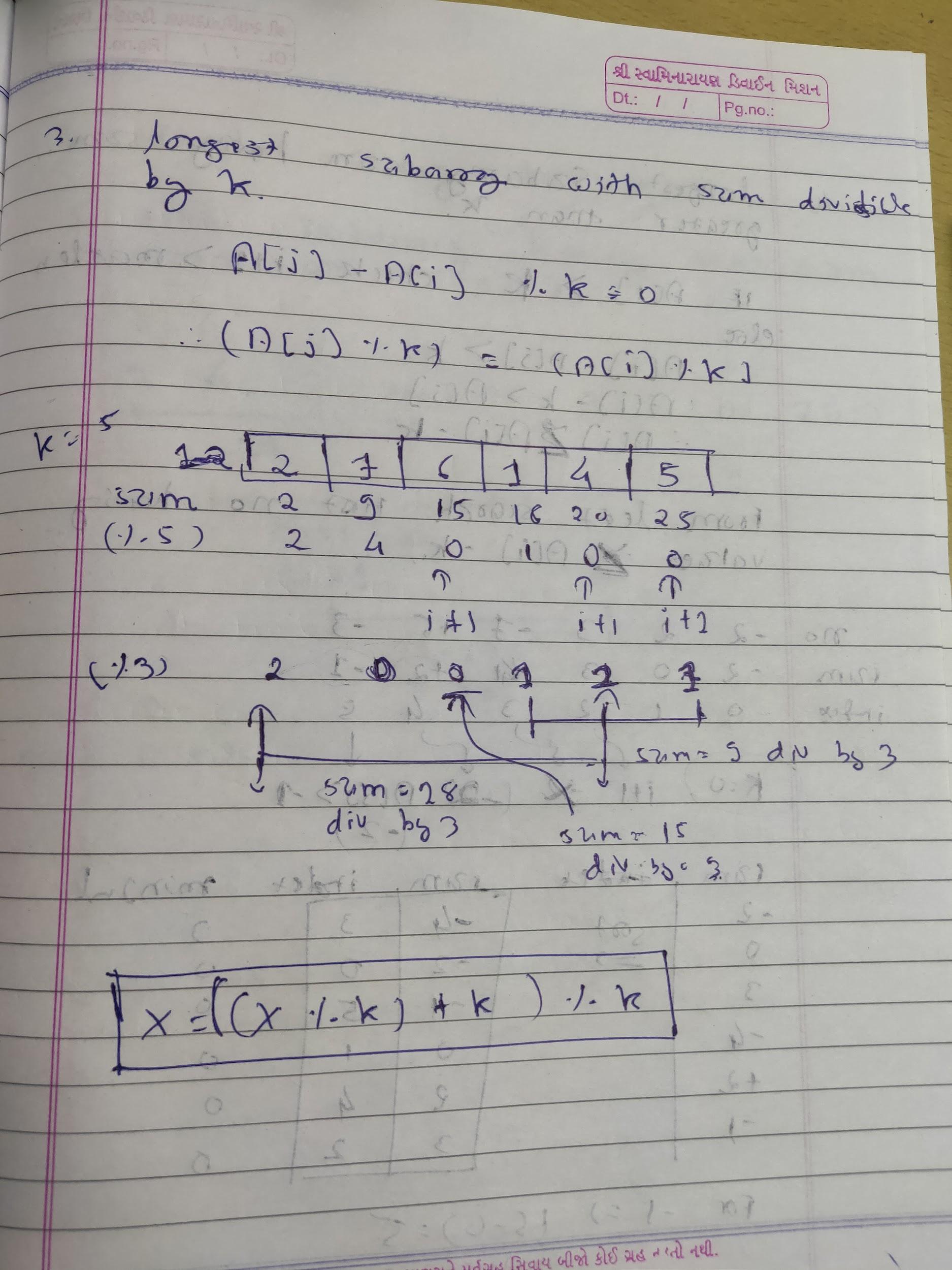
1. Longest Subarray having sum K

<https://www.interviewbit.com/problems/longest-subarray-sum-b/>



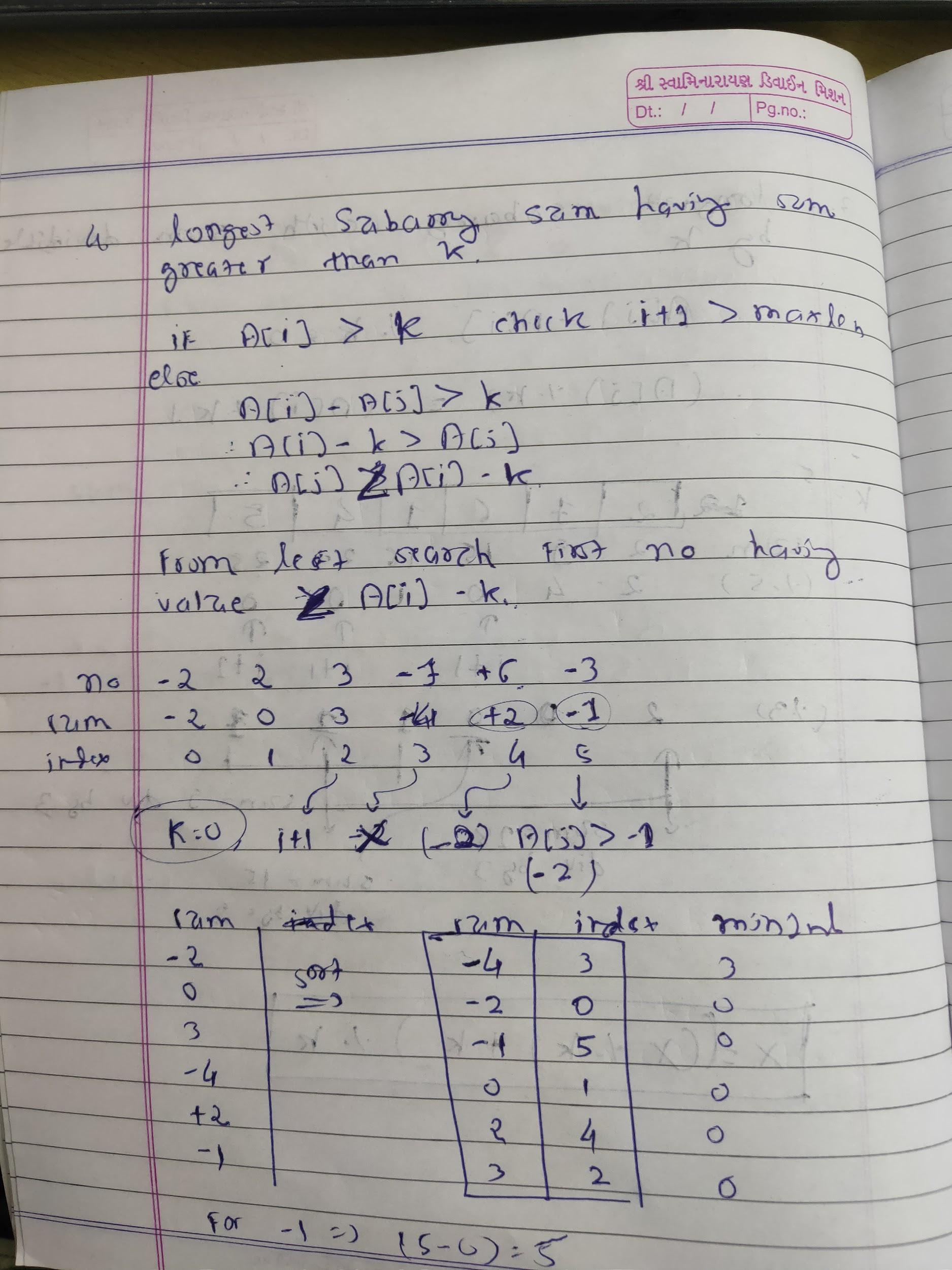
1. Longest Subarray sum divides k

<https://www.geeksforgeeks.org/longest-subarray-sum-divisible-k/>



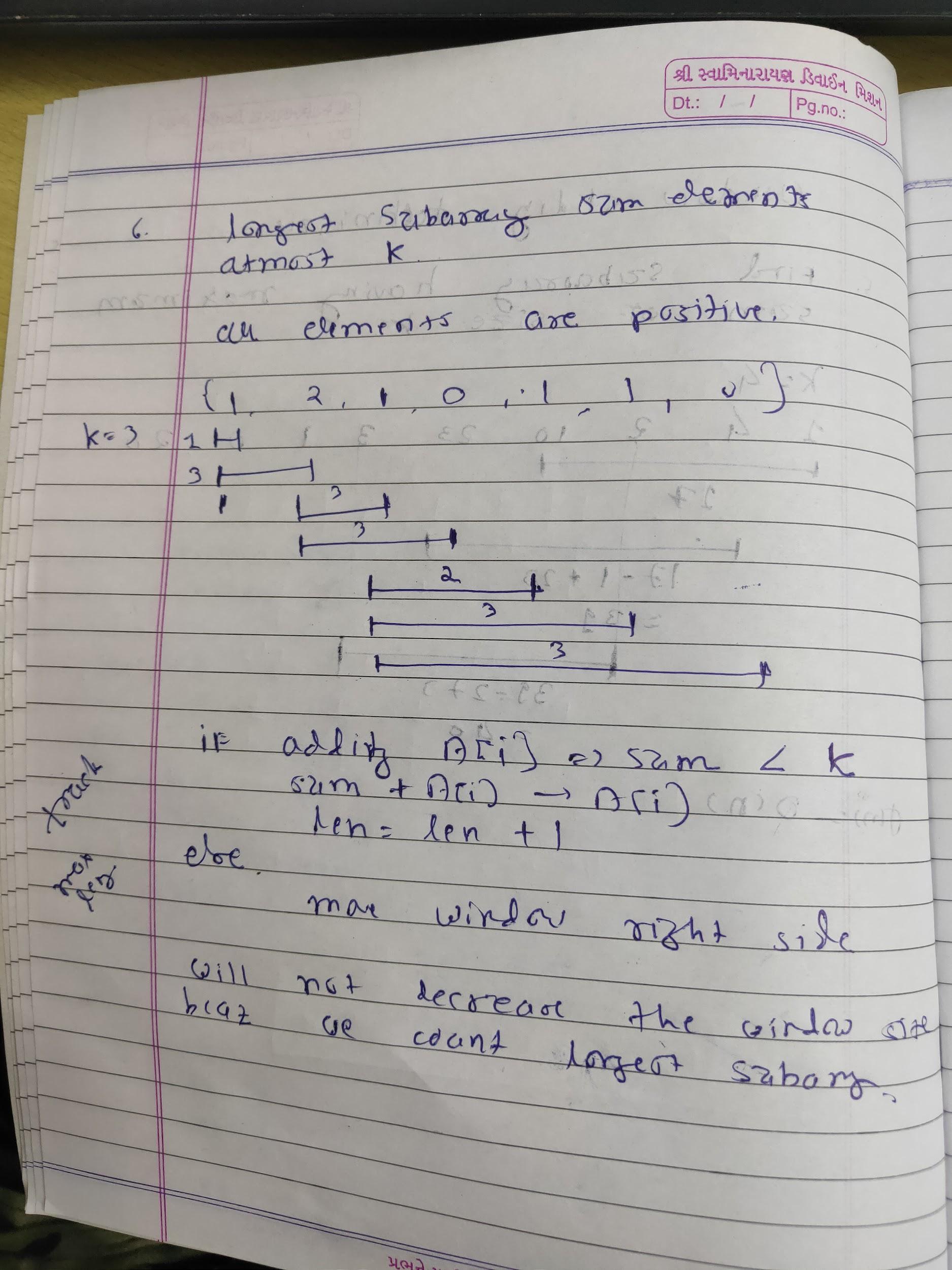
1. Longest Subarray sum having sum greater than k

<https://www.geeksforgeeks.org/longest-subarray-sum-divisible-k/>

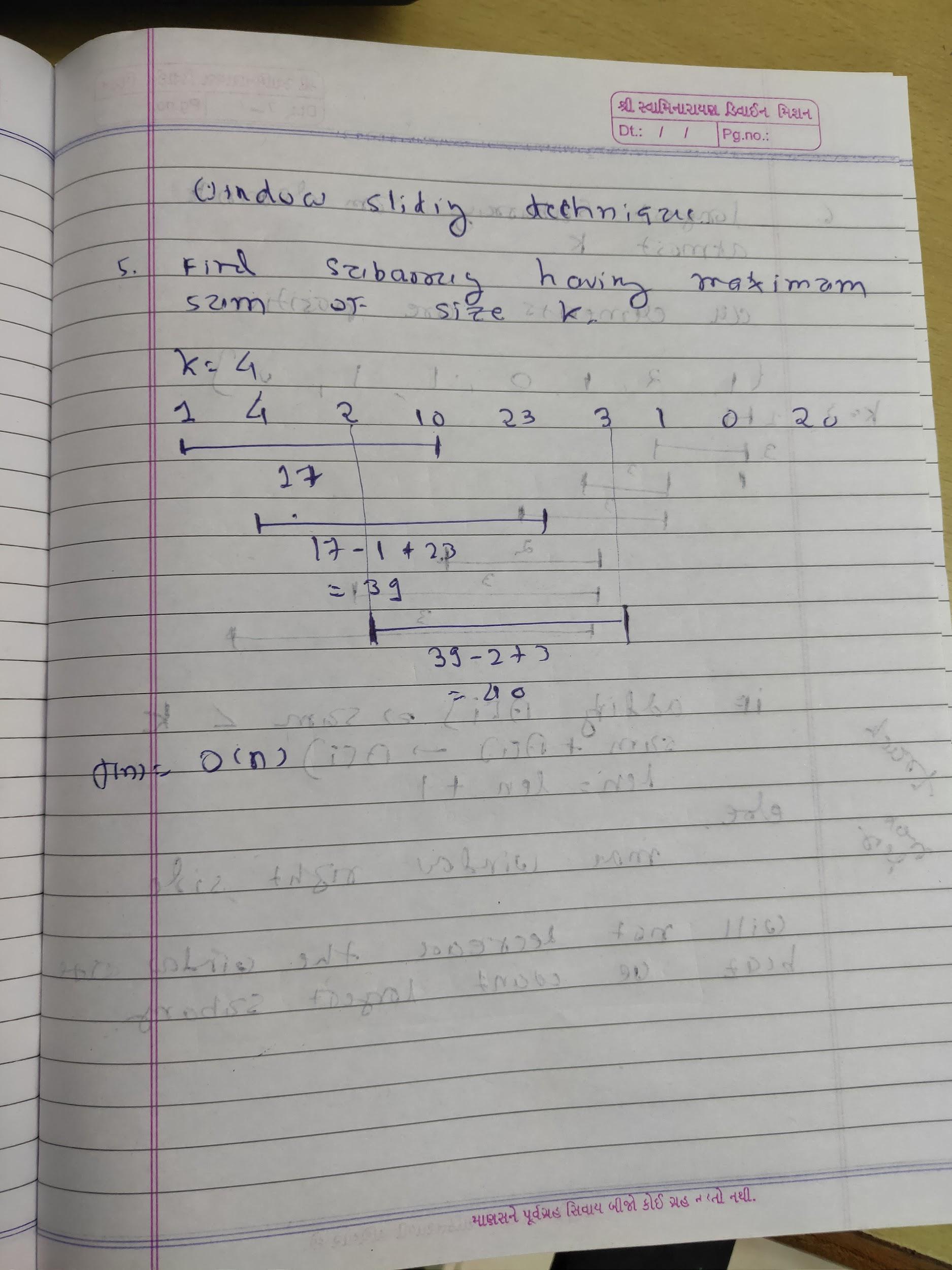


1. Sliding Window
   1. Longest Subarray sum at most k

<https://www.geeksforgeeks.org/longest-subarray-sum-elements-atmost-k/>



* 1. Find subarray of size k having maximum sum
     1. <https://www.geeksforgeeks.org/window-sliding-technique/>



1. Longest Substring without repeating char
   1. <https://leetcode.com/problems/longest-substring-without-repeating-characters/submissions/>
   2. Use Sliding Window
      1. abcdec
      2. abcdec
      3. abcdecb

Maintain a unordered map which maintains char to index mapping

Starting index j and iterate through i

When char is repeated make j = max(first occur index + 1, j)

1. Longest Subarray not having more than k distinct elements
   1. <https://www.geeksforgeeks.org/longest-subarray-not-k-distinct-elements/>
      1. 1 2 3 4 2 3 5 3 5
      2. 1 2 3
      3. 2 3 4 2 3
      4. 2 3 5 3
      5. 2 3 5 3 5
   2. Idea is use unordered map whose size should be always <= k. If more than k inc i and remove ith element from map. Keep track of max len.
   3. i = starting window index , j = ending window index.
2. Maximum in each window of size k
   1. <https://leetcode.com/problems/sliding-window-maximum/submissions/>
   2. <https://www.geeksforgeeks.org/sliding-window-maximum-maximum-of-all-subarrays-of-size-k-using-stack-in-on-time/>
3. Maximum window substring
   1. <https://leetcode.com/problems/minimum-window-substring/submissions/>
   2. Use two pointers i = increment in map, j = decrement in map

Initially size = t.length()

If i increment from 0 then size++

If j decrement to 0 then size--

Check for min len while size = 0

When size is 0 try to reduce window size from front

1. Maximum product subarray
   1. <https://leetcode.com/problems/maximum-product-subarray/submissions/>
   2. int a = maxp;

int b = minp;

maxp = max(nums[i], max(a\*nums[i], b\*nums[i]));

minp = min(nums[i], min(a\*nums[i], b\*nums[i]));

ans = max(ans, maxp);

1. <https://leetcode.com/problems/longest-repeating-character-replacement/>