

A. Problem statement:

From the given Array of integer n, find the maximum length of subarray that are arranged in either even and odd or odd and even.

sample input: [10, 12, 14, 4, 8]

output: 3

Explanation: 14 and 7 and 7 and 8 are arranged in alternate pattern

Expected time complexity as $O(n)$.

Array Equilibrium:

arr[3] = {1, 4, 3, 1, 6}
 total-sum = 17, left-sum = 0
 for(i=0; i<n; i++) {
 total-sum = total-sum - arr[i];
 if (total-sum == left-sum) {
 i = 3
 } else {
 left-sum = left-sum + arr[i];
 }
 }

Kadane's Algorithm:

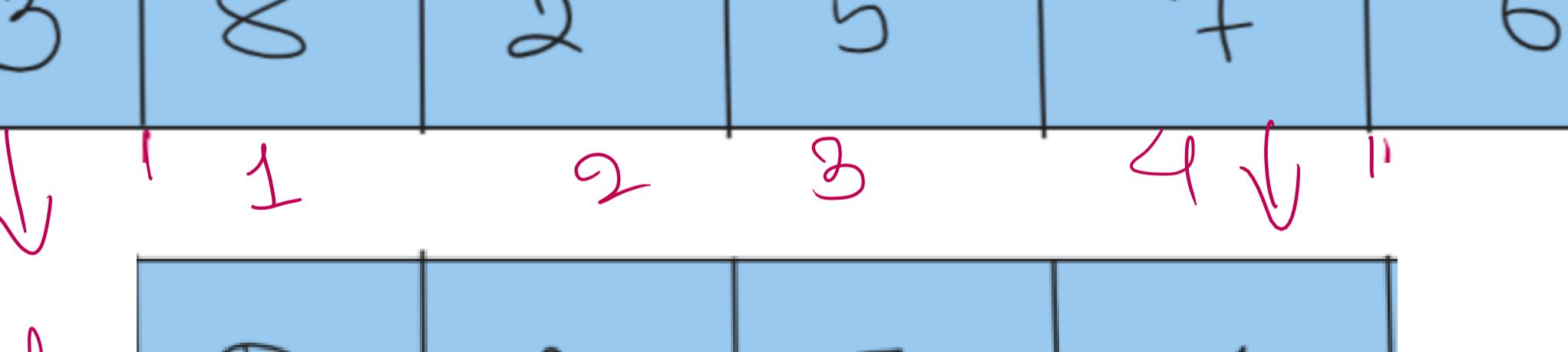
Maximum Sum of Sub-arrays

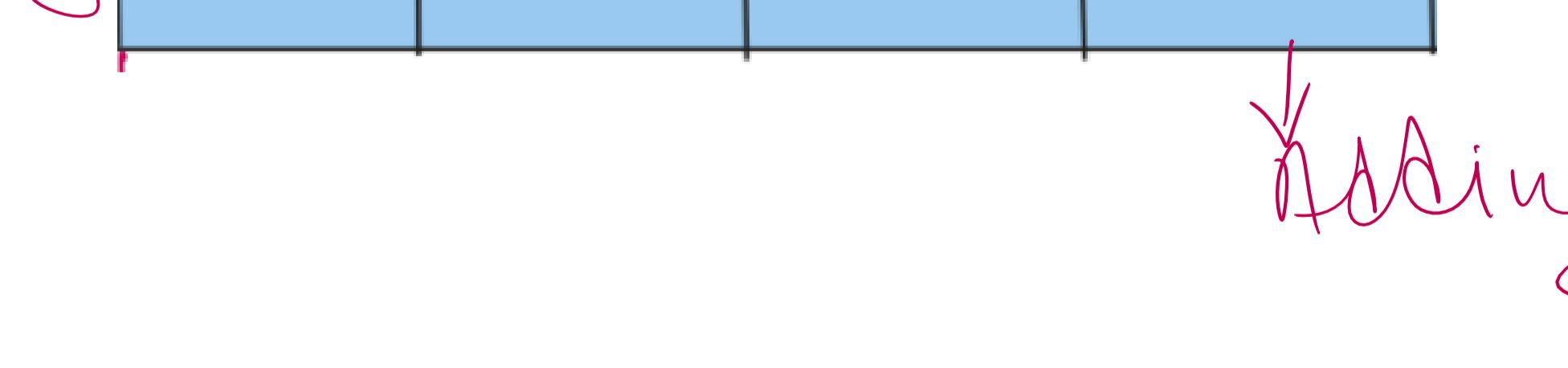
Input1: [1 2 3 -2 5] Input2: [-1 -2 -3 -4]
 Output: 9 Output: -1

Problem statement: From the given Array of N integers find the maximum sum of sub-arrays.

arr[3] = {1, 2, 3, -2, 5} [-1, -2, -3, -4]
 int cur-sum = arr[0], max-sum = arr[0];
 for(i=1; i<n; i++) {
 cur-sum = max(arr[i], cur-sum + arr[i]);
 max-sum = max(cur-sum, max-sum);
 }

Sliding Window Technique


 Window size(w) = 3


 w=4

cur-sum = 0;

for(i=0; i<w; i++) {

cur-sum = cur-sum + arr[i];

3

cur-sum = 18

max-sum = cur-sum;

CurSum = 18

maxSum = 18

Subtracting 3

Adding 7 to the cursum.

for(i=l; i<n-w; i++) {

cur-sum = cur-sum - arr[i-l] + arr[n-w+i];

if (cur-sum > max-sum) {

max-sum = cur-sum;

3