## Analysis

Every contestant should be aware of the standard task in which it is given an array of numbers and it has to be found the subarray with the greatest sum (of course there are negative numbers, otherwise the answer will be the whole array). Here we don't have direct access to the array and we can only ask questions for comparing sums of subarrays.

The first subtask is for 10 points. Here it is expected the most simple idea and understanding of the statement. It is enough to check every two subarrays and to see which will give us the maximal value. The comparison is linear and the subarrays can be with dimensions near the dimensions of the whole array and they are around $\boldsymbol{N}^2$ so the complexity for this subtask is $O(\boldsymbol{N}^3)$.

The second subtask is for 30 points. We have to revise one of the algorithms for solving the problem with the sum. We can view every sum of subarray as a difference of prefix sums. If we fix the right position (in this way we have fixed one of the prefix sums) we have to find the prefix sum (with position lower than the right position), which is the least possible. To determine the least in the beginning we clearly start with prefix sum 0 but in the process we have to check if there is prefix sum less than 0 and this value isn't guaranteed to be in the array to make a direct comparison with the function. Let some prefix subarray has negative sum. If we find its sum with the next element, it will be less than the value of the element. If this sum is greater than or equal to that value, it means that this sum is non-negative. Now we can find the first prefix subarray with negative sum (if such exists) and we can use it as index for the current minimum sum. From now on, it is very easy – every time we compare the previous minimum prefix with the sum of the current prefix and we save the index which gives us the least possible sum. The comparison for the answer is also easy – we just use the given function and save the subarrays with the greatest sum. If we consider that the function for comparison works with linear complexity and we use it linear number of times compared to the number of elements, the complexity here is $O(\boldsymbol{N}^2)$.

The third subtask is for 60 points. Here we have to use more uncommon idea for comparison because the standard is linear on the number times of calls of the function but the function is also linear on the input and we have quadratic complexity. We will use divide and conquer technique. Let we divide the array to left and right part which are almost equal-sized. Let we assume that we have solved the task for each of these arrays. What is the answer to the original problem? Obviously, it will be the answer of the left part, or the answer of the right part or some subarray which includes the middle. This tells us that we will have to expand the task in order to be able to find the maximal subarray (as a sum) which includes the middle. We will save information in each part about the maximal prefix and suffix sum. Now we will be able to find the answer of array if we have the described information for the left and right part. The only thing we have to discuss is if the new pieces of information are easy to be computed. They actually are. If we want to find the maximal prefix sum, it will be equal to the maximal prefix sum in the left part or the sum of the left part plus the maximal prefix sum of the right part. Analogously is for the maximal suffix sum. We have to pay attention that the comparisons which we will conduct are linear on the size of the array. Now we don't have problem and in the worst case, to find the answer we will need two times the size of the whole array, to find the maximal prefix sum we will repeat one and a half times the size of the whole array and for the maximal suffix sum – also one and a half. So on each level we will repeat at

most five times the sizes of the array and the whole procedure will have complexity $O(N \log_2 N)$.

*Author: Iliyan Yordanov*