Kick Start 2019 - Round A

Analysis: Training

To make a fair team, we have to train the members of the team to the same skill level as the most skillful member of the team.
For any **P** students we pick, the time required to make a fair team is = $\Sigma(\max(\mathbf{S}_i, \mathbf{S}_{i+1} \ldots \mathbf{S}_P) - \mathbf{S}_i)$, for all students i = 1 to **P** in the team. Our goal is to minimize this value.
One possible approach could be to go through all possible subsets of **P** students, from the given **N** students. But there exists ${}^{\mathbf{N}}C_{\mathbf{P}}$ such subsets(here symbol C denotes Combination). Number of such subsets will be in the order of Factorial(**N**) and so enumerating through all of them will not fit within the time limit.

Test set 1 (Visible)

We can start with the observation that once we fix the student with highest skill level x, to minimize our goal we should always choose students with skills as high as possible, but less than or equal to x. Hence we can sort students on the basis of skill level in decreasing order, and iterate over each student assuming they would have the highest skill level in the team. Say, this student is at position i in the sorted sequence; the team would be formed of students on positions i, i + 1, ..., i + **P** - 1 (i.e. a contiguous subarray of size **P**).
For each subarray of size **P** in the sorted array, we can calculate the training time required to make a fair team using the aforementioned formula. There are **N** - **P** + 1 subarrays of size **P**, and the complexity of calculating training time of subarray size **P** is O(**P**). So the overall complexity of this approach is O(**N** × **P**), which will be sufficient for test set 1.

Test set 2 (Hidden)

We need to go through all of the subarrays once, but can we calculate the training time of a subarray faster?
Let us assume the array is sorted in decreasing order. The training time formula for a subarray starting at position i is
= $\Sigma(\mathbf{S}[i] - \mathbf{S}[j])$ where j = i to i + **P** -1
= **P** × **S**[i] - $\Sigma(\mathbf{S}[j])$ where j = i to i + **P** - 1
As we always need the sum of a contiguous subarray, we can pre compute the prefix sum of the whole array in advance, and get the sum of any subarray in O(1) time, which makes the calculation of training time O(1).
So, the overall complexity of this approach is O(**N**).