ACM Summer Challenge 2020

Array Manipulation

Editorial

K Times Array

K Times Array was more of a mathematical question. Still, it had little to do with array manipulation. Consider a position i in the array. Now for this position **i**, you need to find the count of all integers from $\mathbf{A_{i+1}}$ to $\mathbf{A_{N-1}}$ which are less than $\mathbf{A_i}$. Let us call this **count1**.
Also, you need to find the count of all integers from $\mathbf{A_0}$ to $\mathbf{A_{i-1}}$ which are less than $\mathbf{A_i}$. Let us call this **count2**.
Now let us observe the array B which is the concatenation of A into itself K times.
Let's observe the last A concatenated into B.
Let's find all those pairs which can be formed using count1.
For $\mathbf{A_i}$ in this **last** array A, it has count1 number of pairs.
For **second last** array A in B, for position **i** in this second last array, it will have **2 * count1** number of required pairs!(Check using your own example)
Similarly, for the **first** array A in B, this position **i** will have **K*count1** pairs.
Thus, using **count1**, for $\mathbf{i^{th}}$ position, we get required pairs which are:

$$\frac{K \times (K+1) \times count1}{2}$$

Now let's find all those pairs which can be formed using count2.

Again, observe the **last** array A, it will have **0** pairs because all the elements are before position **i**.
But for the **second last** array A, it will have **count2** pairs possible (the ones that lie in the last array will make pairs with $\mathbf{i^{th}}$ position for this array).
For **third last** array A, it will have **2 * count2** pairs possible.
Similarly, for the **first** array A in B, this position **i** will have **(K-1) * count2** pairs.
Thus, using **count2**, for $\mathbf{i^{th}}$ position, we get required pairs which are:

$$\frac{K \times (K-1) \times count2}{2}$$

So for any position **i**, the sum of required pairs will be the sum of above two expressions.

You need to find the overall sum of required pairs for all positions where **i** belongs to **0** to **N-1**.
This can be done in **$O(N^2)$** using nested loops.

My Team Always Wins

This question is an excellent example of array manipulation and linear traversal. You need to find such a segment of **K** minutes in which Monica can clean maximum items while being awake, which she was skipping due to being asleep.
You can use a variable for storing the sum of items cleaned by her when she was already awake. Let us call this variable **sum**.
Another variable can be used for maintaining a maximum of items cleaned by her in **K** minutes being looped from **i = 1** to **i = N - K + 1** considering **K** minutes from **i** to **i + K -1** and only those items need to be added to this variable which are cleaned when she is asleep. Let us call this variable **sum1**.
Your answer will then be **sum + sum1**.
This can also be implemented using Prefix Array. Head over here for more information.
This can be done in **O(N)**.

Under Attack

If you observe the question carefully, a soldier with a strength higher than the strength of his/her neighbours can never have his/her strength reduced to **0**.
Hence, all you need to find is the sum of the strengths of such soldiers whose strength is greater than their neighbours, which can be done in **O(NxM)** using nested loops.