Problem name: Mascot Song

Language: English

Solution

This problem was all about smart simulation with the help of the simple data structure – an array. Using the definition of a block, it directly follows that some block ends at position i and the next one starts at position $i+1$ if and only if $A_i \geq A_{i+1}$. Now, for all $1 \leq i < n$, let us call a consecutive pair $(i, i+1)$ **bad** if $A_i \geq A_{i+1}$ and let $b(A)$ – the total number of bad pairs of sequence A. Now, the most importan (and obvious) fact is that the number of blocks of sequence A equals $1 + b(A)$.

The solution for 30 points is to simulate both types of queries directly, with constant complexity for query of type 1 and linear complexity for query of type 2. Number of bad pairs after each query can be computed in linear time and the time complexity of this algorithm is $O(q \cdot n)$.

For the further 30 points (only queries of type 1) it should be noted that query "1 x y" can only affect consecutive pairs $(x-1, x)$ and $(x, x+1)$, if they exists. Therefore, if we precompute $b(A)$ at the beginning, we can check in constant time if $b(A)$ changed and how; after that, we update $b(A)$ and A_x. Time complexity of this solution is $O(n+q)$.

For the full score, we need to handle the queries of type 2. However, note that the query "2 z" has the same effect as switching first z elements with last $n-z$ elements. During this process, we "break" one consecutive pair - (A_z, A_{z+1}) and form another - (A_n, A_1). The rest $n-3$ consecutive pairs stay the same; therefore, we can update $b(A)$ in constant time. In order to keep up with the indices, we will have one extra variable – $first$ which will denote the original index of the currently first element of sequence A; for example, if "2 z" is the first query, the sequence would look like $(A_{z+1}, A_{z+2}, \ldots, A_n, A_1, A_2, \ldots A_z)$ and we would have $first = z+1$. With this notation, the current position x is actually the position $first + x - 1$ of the original array (all operations are done modulo n). Time complexity of this solution is $O(n+q)$ and memory complexity is $O(n)$.