cover

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	512 megabytes

You are given an array $a_{1,2,\cdots,n}$ of length n and m modification operations. The i-th operation is described by three parameters l_i, r_i, c_i, indicating that you may choose to set all elements in the subarray $a_{l_i}, a_{l_i+1}, \cdots, a_{r_i}$ to c_i. For each operation, you can choose to either apply it or skip it.

Your goal is to maximize the number of contiguous segments in the final array by optimally deciding whether to apply each operation (processed in the given order). The number of contiguous segments is defined as $1 + \sum_{i=2}^{n}[a_i \neq a_{i-1}]$, which counts the number of adjacent distinct elements plus one.

Additionally, all operations satisfy the following special property: for any two operations $i < j$, the interval $[l_j, r_j]$ is not a subinterval of $[l_i, r_i]$. In other words, there are no $i < j$ such that both $l_i \leq l_j$ and $r_j \leq r_i$ hold simultaneously.

Input

The input consists of multiple test cases. The first line contains an integer $T(1 \leq T \leq 10000)$, denoting the number of test cases.

Each test case consists of the following parts:

- The first line contains two positive integers $n(1 \leq n \leq 3 \times 10^5)$ and $m(1 \leq m \leq 3 \times 10^5)$, denoting the length of the array and the number of operations, respectively.
- The second line contains n positive integers $a_1, a_2, \cdots, a_n$ $(1 \leq a_i \leq n)$, representing the initial array.
- The next m lines each contain three integers l_k, r_k, c_k, describing the k-th operation $(1 \leq l_k \leq r_k \leq n$, $1 \leq c_k \leq n)$.

It is guaranteed that the sum of n across all test cases does not exceed 3×10^5, and the sum of m does not exceed 3×10^5.

Output

Output a single integer, representing the maximum number of contiguous segments achievable after optimally applying the operations.

Example

standard input	standard output
2 5 5 1 1 1 1 1 1 2 3 2 3 4 1 3 4 2 4 4 5 5 3 3 1 1 1 1 2 2 2	4 3