An integer M and a non-empty zero-indexed array A consisting of N non-negative integers are given. All integers in array A are less than or equal to M.

A pair of integers (P, Q), such that 0 ≤ P ≤ Q < N, is called a *slice* of array A. The slice consists of the elements A[P], A[P + 1], ..., A[Q]. A *distinct slice* is a slice consisting of only unique numbers. That is, no individual number occurs more than once in the slice.

For example, consider integer M = 6 and array A such that:

A[0] = 3

A[1] = 4

A[2] = 5

A[3] = 5

A[4] = 2

There are exactly nine distinct slices: (0, 0), (0, 1), (0, 2), (1, 1), (1, 2), (2, 2), (3, 3), (3, 4) and (4, 4).

The goal is to calculate the number of distinct slices.

Write a function:

class Solution { public int solution(int M, int[] A); }

that, given an integer M and a non-empty zero-indexed array A consisting of N integers, returns the number of distinct slices.

If the number of distinct slices is greater than 1,000,000,000, the function should return 1,000,000,000.

For example, given integer M = 6 and array A such that:

A[0] = 3

A[1] = 4

A[2] = 5

A[3] = 5

A[4] = 2

the function should return 9, as explained above.

Assume that:

* N is an integer within the range [1..100,000];
* M is an integer within the range [0..100,000];
* each element of array A is an integer within the range [0..M].

Complexity:

* expected worst-case time complexity is O(N);
* expected worst-case space complexity is O(M), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.