

AVAILABLE
LESSONS:*Lesson 1*
Iterations*Lesson 2*
Arrays*Lesson 3*
Time Complexity*Lesson 4*
Counting
Elements*Lesson 5*
Prefix Sums*Lesson 6*
Sorting*Lesson 7*
Stacks and
Queues*Lesson 8*
Leader*Lesson 9*
Maximum slice
problem*Lesson 10*
Prime and
composite
numbers*Lesson 11*

PAINLESS

CountDistinctSlices

START

Count the number of distinct slices (containing only unique numbers).

Programming language: C++ ▼

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 $0 \leq P \leq Q < N$, is called a *slice* of array A . The slice consists of the elements $A[P], A[P + 1], \dots, A[Q]$. 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:

```
int solution(int M, vector<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
```

Sieve of
Eratosthenes

Lesson 12

Euclidean
algorithm

Lesson 13

Fibonacci
numbers

Lesson 14

Binary search
algorithm

Lesson 15

**Caterpillar
method**

Lesson 16

Greedy
algorithms

Lesson 17

Dynamic
programming

Lesson 90

Tasks from
Indeed Prime
2015 challenge

Lesson 91

Tasks from
Indeed Prime
2016 challenge

Lesson 92

Tasks from
Indeed Prime
2016 College
Coders
challenge

Lesson 99

$A[2] = 5$

$A[3] = 5$

$A[4] = 2$

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.

Copyright 2009–2016 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

**For
programmers**

Lessons Challenges
Terms FAQ

For companies

Tour About
Pricing us
Blog Jobs
Privacy Terms
Cookies
API

**Sign up for our
newsletter:**

Information about
upcoming challenges,
solutions and lessons
directly in your inbox.

Social:

[f](#) [t](#) [in](#)

Contact us:

For customer support
queries:

UK +44 (0) 208 970
78 68

US 1-415-466-8085
support@codility.com

For sales queries:

UK +44 (0) 208 970
78 67

US 1-415-466-8085
sales@codility.com

© 2009–2016 Codility Ltd., registered in England and Wales (No. 7048726). VAT ID
GB981191408. Registered office: 107 Cheapside, London EC2V 6DN