Codility_

## Candidate Report: trainingPNPVVA-QZQ

Check out Codility training tasks

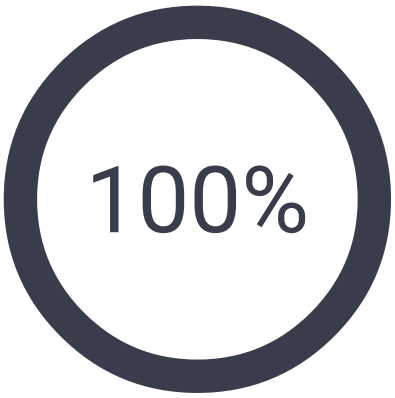
Test Name:

Summary Timeline

### Tasks summary

| Task | Time spent | Score |
|---|---|---|
| **Distinct** C | 19 min | 100% |

### Total score




## Tasks Details

Easy

1. **Distinct**
Compute number of distinct values in an array.

| Task Score | Correctness | Performance |
|---|---|---|
| 100% | 100% | 100% |

### Task description

Write a function

```
int solution(int A[], int N);
```

that, given an array A consisting of N integers, returns the number of distinct values in array A.

For example, given array A consisting of six elements such that:

```
A[0] = 2    A[1] = 1    A[2] = 1
A[3] = 2    A[4] = 3    A[5] = 1
```

the function should return 3, because there are 3 distinct values appearing in array A, namely 1, 2 and 3.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [0..100,000];

### Solution

| | |
|---|---|
| Programming language used: | C |
| Total time used: | 19 minutes |
| Effective time used: | 19 minutes |
| Notes: | *not defined yet* |

### Task timeline

- each element of array A is an integer within the range [−1,000,000..1,000,000].



10:35:56 10:54:51

Code: 10:54:50 UTC, c, final, score: **100** show code in pop-up

```c
// you can write to stdout for debugging purposes, e
// printf("this is a debug message\n");

#define OFFSET 1000000
int solution(int A[], int N) {
    // write your code in C99 (gcc 6.2.0)
    int i, ret = 0;
    int ARR[2000000+1] = {0};
    for (i = 0; i < N; i++){
        ARR[A[i]+OFFSET] = 1;
    }
    for (i = 0; i <2000000+1; i++){
        if (ARR[i])
            ret+=1;
    }
    return ret;
}
```

## Analysis summary

The solution obtained perfect score.

## Analysis

Detected time complexity: **O(N*log(N)) or O(N)**

| expand all | Example tests | |
|---|---|---|
| ▶ | example1<br>example test, positive answer | ✓ OK |
| expand all | **Correctness tests** | |
| ▶ | extreme_empty<br>empty sequence | ✓ OK |
| ▶ | extreme_single<br>sequence of one element | ✓ OK |
| ▶ | extreme_two_elems<br>sequence of three distinct elements | ✓ OK |
| ▶ | extreme_one_value<br>sequence of 10 equal elements | ✓ OK |
| ▶ | extreme_negative<br>sequence of negative elements, length=5 | ✓ OK |
| ▶ | extreme_big_values<br>sequence with big values, length=5 | ✓ OK |
| ▶ | | |

| | | |
|---|---|---|
| | medium1<br>chaotic sequence of value sfrom [0..1K], length=100 | ✓ OK |
| ▶ | medium2<br>chaotic sequence of value sfrom [0..1K], length=200 | ✓ OK |
| ▶ | medium3<br>chaotic sequence of values from [0..10], length=200 | ✓ OK |

expand all **Performance tests**

| | | |
|---|---|---|
| ▶ | large1<br>chaotic sequence of values from [0..100K], length=10K | ✓ OK |
| ▶ | large_random1<br>chaotic sequence of values from [-1M..1M], length=100K | ✓ OK |
| ▶ | large_random2<br>another chaotic sequence of values from [-1M..1M], length=100K | ✓ OK |

---