Codility_

Candidate Report: trainingXD2PFN-6P6

Check out Codility training tasks

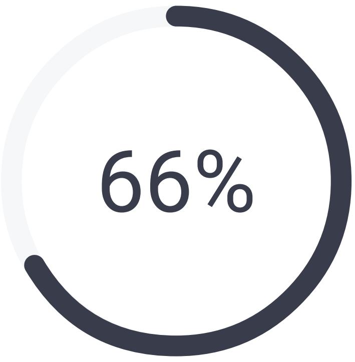
Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
OddOccurrencesInArray ⚠ Java 8	44 min	66%

Total score



Identity verification
Add another layer of security with Identity Verification, learn more.

Tasks Details

Easy	1. OddOccurrencesInArray Find value that occurs in odd number of elements.	Task Score	Correctness	Performance
		66%	100%	25%

Task description

A non-empty array A consisting of N integers is given. The array contains an odd number of elements, and each element of the array can be paired with another element that has the same value, except for one element that is left unpaired.

For example, in array A such that:

```
A[0] = 9  A[1] = 3  A[2] = 9
A[3] = 3  A[4] = 9  A[5] = 7
A[6] = 9
```

- the elements at indexes 0 and 2 have value 9,
- the elements at indexes 1 and 3 have value 3,
- the elements at indexes 4 and 6 have value 9,
- the element at index 5 has value 7 and is unpaired.

Write a function:

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

that, given an array A consisting of N integers fulfilling the above conditions, returns the value of the unpaired element.

For example, given array A such that:

```
A[0] = 9  A[1] = 3  A[2] = 9
A[3] = 3  A[4] = 9  A[5] = 7
A[6] = 9
```

the function should return 7, as explained in the example above.

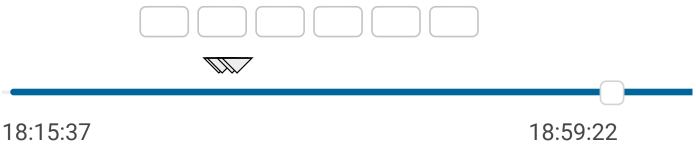
Write an **efficient** algorithm for the following assumptions:

- N is an odd integer within the range [1..1,000,000];

Solution

Programming language used:	Java 8
Total time used:	44 minutes ?
Effective time used:	44 minutes ?
Notes:	*not defined yet*

Task timeline ?



Code: 18:59:22 UTC, java, final, score: **66**

show code in pop-up

```
// you can also use imports, for example:
import java.util.*;

// you can write to stdout for debugging purposes,
// System.out.println("this is a debug message");

class Solution {
    public int solution(int[] A) {
        int i,k,j,c=0;
        int l=A.length;
        Set<Integer> s=new HashSet<Integer>();
```

- each element of array A is an integer within the range [1..1,000,000,000];
- all but one of the values in A occur an even number of times.

```
        List<Integer> list1=new ArrayList<Integer>(

        for(i=0;i<l;i++){
            if(!s.contains(A[i])){
                s.add(A[i]);
            }
            else{
                list1.add(A[i]);
            }
        }
        for(int a:s){
            for(int b:list1){
                if(a==b){
                    c++;

                }
            }
            if(c%2!=0){
                c=0;
            }
            else{
                return a;
            }
        }
        return A[i];
        // write your code in Java SE 8
    }
}
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

Detected time complexity: **O(N**2)**

expand all	Example tests	
►	example1 example test	✓ OK
expand all	Correctness tests	
►	simple1 simple test n=5	✓ OK
►	simple2 simple test n=11	✓ OK
►	extreme_single_item [42]	✓ OK
►	small1 small random test n=201	✓ OK
►	small2 small random test n=601	✓ OK
expand all	Performance tests	
►	medium1 medium random test n=2,001	✓ OK
►	medium2 medium random test n=100,003	✗ TIMEOUT ERROR Killed. Hard limit reached: 7.000 sec.
►	big1 big random test n=999,999, multiple repetitions	✗ TIMEOUT ERROR Killed. Hard limit reached: 14.000 sec.
►	big2 big random test n=999,999	✗ TIMEOUT ERROR Killed. Hard limit reached: 19.000 sec.

The PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.