Test results - Codility

Test results - Codility

Test results - Codility

Test results - Codility

Test results - Codility

Test results - Codility

Test results - Codility

Test results - Codility

Test results - Codility

Codility_

## CodeCheck Report: trainingHURCA9-U92

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

### Tasks summary

| Task | Time spent | Score |
|---|---|---|
| ScooterRoad<br>C++ | 6 min | 100% |

### Total score

100%

## Tasks Details

### 1. ScooterRoad

Medium

Calculate the minimum time that you need to get through the diversified road to your work.

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

### Task description

You have to be at your work as soon as possible. The road on your route to work may consist of two types of surface: asphalt or sand. To simplify the description, it will be denoted by a string R consisting only of the letters: "A" for an asphalt segment and "S" for a sand segment. All segments represent the same distance. For example, R = "SAAS" describes a road comprising of sand, asphalt, asphalt and sand segments.

When you go on foot, you need 20 minutes to pass through an asphalt segment and 30 minutes through a sand segment. You also have an electric scooter, which needs 5 minutes to pass through an asphalt segment and 40 minutes through a sand segment.

You start your journey on the scooter, but at any point you can get off the scooter and go on foot for the rest of the journey. What is the shortest time in which you can get to work?

Write a function:

```
int solution(string &R);
```

that, given a string R of length N, representing the road to work, returns the minimum time that you need to get to work.

### Solution

| | |
|---|---|
| Programming language used: | C++ |
| Total time used: | 6 minutes |
| Effective time used: | 6 minutes |
| Notes: | *not defined yet* |

### Task timeline

11:36:54 11:42:04

Code: 11:42:04 UTC, cpp, final, score: **100**

show code in pop-up

```
// you can use includes, for example:
// #include <algorithm>
```

**Examples:**

1. Given R = "ASAASS", your function should return 115. You ride on the scooter over the first four segments ("ASAA") in 5 + 40 + 5 + 5 = 55 and then you go on foot through "SS" in 30 + 30 = 60. Altogether, your journey will take 55 + 60 = 115.

2. Given R = "SSA", the function should return 80. You do not ride on the scooter at all, and you go on foot in 30 + 30 + 20 = 80.

3. Given R = "SSSSAAA", the function should return 175. You ride on the scooter all the time in 40 + 40 + 40 + 40 + 5 + 5 + 5 = 175.

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

- N is an integer within the range [1..100,000];
- string R is made only of the characters 'S' and/or 'A'.

```cpp
#include <vector>

// you can write to stdout for debugging purposes,
// cout << "this is a debug message" << endl;

int cost(bool scooter,bool sand){
    vector<vector<int>> costs = {{20,30},{5,40}};
    return costs[scooter][sand];
}

int solution(string &R) {
   int n = R.size();
   vector<int> foot(n+1,0);
   for(int i=n-1;i>=0;i--){
       foot[i] = foot[i+1] + cost(false,R[i]=='S')
   }
   int ans = foot[0];
   int c = 0;
   for(int i=0;i<n;i++){
       c+=cost(true,R[i]=='S');
       ans=min(ans,c+ foot[i+1]);
   }
   return ans;
}
```

## Analysis summary

The solution obtained perfect score.

## Analysis

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

expand all **Example tests**

▶ example1 ✓ OK
First example test.

▶ example2 ✓ OK
Second example test.

▶ example3 ✓ OK
Third example test.

expand all **Correctness tests**

▶ very_short_road ✓ OK
N = 1.

▶ short_road ✓ OK
N <= 3.

▶ only_scooter ✓ OK
Only scooter is used.

▶ only_walking ✓ OK
Scooter is not used at all.

▶ all_asphalt_first ✓ OK
Road can be described by "AA...ASS...S". N <= 200.

▶ small_random_change_point ✓ OK
Getting off the scooter at a random position. N <= 200.

expand all **Performance tests**

▶ medium_random ✓ OK
Medium random tests. N <= 10,000.

▶

| | | |
|---|---|---|
| | medium_random_change_point<br>Medium tests, getting off the scooter at a random position. N <= 10,000. | ✓ OK |
| ▶ | big_random<br>Big random tests. | ✓ OK |
| ▶ | big_random_change_point<br>Big tests, getting off the scooter at a random position. | ✓ OK |
| ▶ | big_corner_cases<br>Big tests with corner cases. | ✓ OK |

Codility_

# CodeCheck Report: trainingWE98ZF-T6D

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

## Tasks summary

| Task | Time spent | Score |
|---|---|---|
| TapeEquilibrium<br>C# ⚠ | 5 min | 100% |

## Total score

100%

## Tasks Details

| Easy | 1. TapeEquilibrium<br>Minimize the value \|(A[0] + ... + A[P-1]) - (A[P] + ... + A[N-1])\|. | Task Score | Correctness | Performance |
|---|---|---|---|---|
| | | 100% | 100% | 100% |

### Task description

A non-empty array A consisting of N integers is given. Array A represents numbers on a tape.

Any integer P, such that 0 < P < N, splits this tape into two non-empty parts: A[0], A[1], ..., A[P − 1] and A[P], A[P + 1], ..., A[N − 1].

The *difference* between the two parts is the value of: |(A[0] + A[1] + ... + A[P − 1]) − (A[P] + A[P + 1] + ... + A[N − 1])|

In other words, it is the absolute difference between the sum of the first part and the sum of the second part.

For example, consider array A such that:

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

We can split this tape in four places:

- P = 1, difference = |3 − 10| = 7
- P = 2, difference = |4 − 9| = 5
- P = 3, difference = |6 − 7| = 1
- P = 4, difference = |10 − 3| = 7

### Solution

| | |
|---|---|
| Programming language used: | C# |
| Total time used: | 5 minutes |
| Effective time used: | 5 minutes |
| Notes: | *not defined yet* |

### Task timeline

09:50:14 09:54:27

Code: 09:54:27 UTC, cs, final, score: **100**

show code in pop-up

```
using System;
// you can also use other imports, for example:
// using System.Collections.Generic;
```

Write a function:

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

that, given a non-empty array A of N integers, returns the minimal difference that can be achieved.

For example, given:

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

the function should return 1, as explained above.

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

- N is an integer within the range [2..100,000];
- each element of array A is an integer within the range [−1,000..1,000].

```

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

class Solution {
    public int solution(int[] A) {
        int l = A[0],r = A[1];
        for (var i =1;++i <A.Length; r += A[i]);
        int min = Math.Abs(l-r);
        for(var P =1;++P<A.Length;){
            int sum = Math.Abs((l += A[P-1])-(r-=A[
            if(sum<min)
                 min = sum;
        }
        return min;
    }
}
```

## Analysis summary

The solution obtained perfect score.

## Analysis

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

expand all **Example tests**

▶ example ✓ OK
example test

expand all **Correctness tests**

▶ double ✓ OK
two elements

▶ simple_positive ✓ OK
simple test with positive numbers, length = 5

▶ simple_negative ✓ OK
simple test with negative numbers, length = 5

▶ simple_boundary ✓ OK
only one element on one of the sides

▶ small_random ✓ OK
random small, length = 100

▶ small_range ✓ OK
range sequence, length = ~1,000

▶ small ✓ OK
small elements

expand all **Performance tests**

▶ medium_random1 ✓ OK
random medium, numbers from 0 to 100, length = ~10,000

▶ medium_random2 ✓ OK
random medium, numbers from -1,000 to 50, length = ~10,000

▶ large_ones ✓ OK
large sequence, numbers from -1 to 1, length = ~100,000

▶

| | |
|---|---|
| large_random<br>random large, length = ~100,000 | ✓ OK |
| ▶ large_sequence<br>large sequence, length = ~100,000 | ✓ OK |
| ▶ large_extreme<br>large test with maximal and minimal values, length = ~100,000 | ✓ OK |

Codility_

## CodeCheck Report: training8EZDNW-6ZN

Test Name:

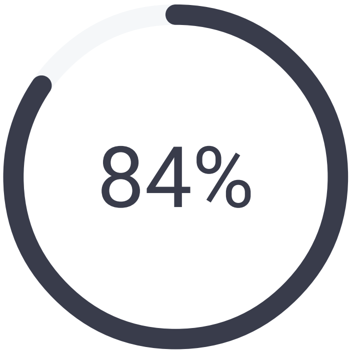
Check out Codility training tasks

Summary Timeline AI Assistant Transcript

### Tasks summary

| Task | Time spent | Score |
|---|---|---|
| TapeEquilibrium<br>C# ⚠ | 8 min | 84% |

### Total score



### Tasks Details

Easy

1. TapeEquilibrium

Minimize the value |(A[0] + ... + A[P-1]) - (A[P] + ... + A[N-1])|.

| Task Score | Correctness | Performance |
|---|---|---|
| 84% | 71% | 100% |

### Task description

A non-empty array A consisting of N integers is given. Array A represents numbers on a tape.

Any integer P, such that 0 < P < N, splits this tape into two non-empty parts: A[0], A[1], ..., A[P − 1] and A[P], A[P + 1], ..., A[N − 1].

The *difference* between the two parts is the value of: |(A[0] + A[1] + ... + A[P − 1]) − (A[P] + A[P + 1] + ... + A[N − 1])|

In other words, it is the absolute difference between the sum of the first part and the sum of the second part.

For example, consider array A such that:

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

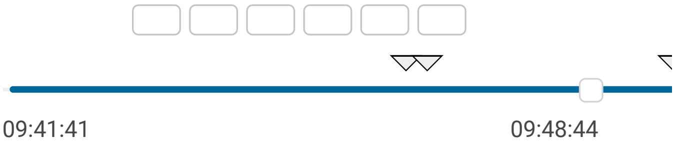
We can split this tape in four places:

- P = 1, difference = |3 − 10| = 7
- P = 2, difference = |4 − 9| = 5
- P = 3, difference = |6 − 7| = 1
- P = 4, difference = |10 − 3| = 7

### Solution

| | |
|---|---|
| Programming language used: | C# |
| Total time used: | 8 minutes ? |
| Effective time used: | 8 minutes ? |
| Notes: | *not defined yet* |

### Task timeline ?



Code: 09:48:43 UTC, cs, final, score: **84**

show code in pop-up

```
using System;
// you can also use other imports, for example:
// using System.Collections.Generic;
```

Write a function:

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

that, given a non-empty array A of N integers, returns the minimal difference that can be achieved.

For example, given:

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

the function should return 1, as explained above.

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

- N is an integer within the range [2..100,000];
- each element of array A is an integer within the range [−1,000..1,000].

```

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

class Solution {
    public int solution(int[] A) {
       int len = A.Length;
       int min = Int32.MaxValue;
       int sum = 0;
       int suma =0;
       foreach(var i in A)
            sum += i;
       for (int i = 0;i<len;i++){
           suma +=A[i];
           int a =(int)Math.Abs(suma - (sum-suma))
           if(a<min)
                min = a;
       }
    return min;
    }
}
```

## Analysis summary

The following issues have been detected: wrong answers.

For example, for the input `[-1000, 1000]` the solution returned a wrong answer (got 0 expected 2000).

## Analysis

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

| expand all | Example tests |
|---|---|
| ▶ example<br>example test | ✓ OK |

| expand all | Correctness tests |
|---|---|
| ▶ double<br>two elements | ✗ WRONG ANSWER<br>got 0 expected 2000 |
| ▶ simple_positive<br>simple test with positive numbers, length = 5 | ✓ OK |
| ▶ simple_negative<br>simple test with negative numbers, length = 5 | ✓ OK |
| ▶ simple_boundary<br>only one element on one of the sides | ✓ OK |
| ▶ small_random<br>random small, length = 100 | ✓ OK |
| ▶ small_range<br>range sequence, length = ~1,000 | ✓ OK |
| ▶ small<br>small elements | ✗ WRONG ANSWER<br>got 0 expected 20 |

| expand all | Performance tests |
|---|---|
| ▶ medium_random1<br>random medium, numbers from 0 to 100, length = ~10,000 | ✓ OK |
| ▶ medium_random2<br>random medium, numbers from -1,000 | ✓ OK |

| | |
|---|---|
| to 50, length = ~10,000 | |
| ▶ large_ones<br>large sequence, numbers from -1 to 1, length = ~100,000 | ✓ OK |
| ▶ large_random<br>random large, length = ~100,000 | ✓ OK |
| ▶ large_sequence<br>large sequence, length = ~100,000 | ✓ OK |
| ▶ large_extreme<br>large test with maximal and minimal values, length = ~100,000 | ✓ OK |

Codility_

## CodeCheck Report: trainingWK64SU-SP9

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

### Tasks summary

| Task | Time spent | Score |
|---|---|---|
| PermMissingElem<br>C# ⚠ | 5 min | 100% |

### Total score

100%

## Tasks Details

Easy

### 1. PermMissingElem

Find the missing element in a given permutation.

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

### Task description

An array A consisting of N different integers is given. The array contains integers in the range [1..(N + 1)], which means that exactly one element is missing.

Your goal is to find that missing element.

Write a function:

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

that, given an array A, returns the value of the missing element.

For example, given array A such that:

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

the function should return 4, as it is the missing element.

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

- N is an integer within the range [0..100,000];
- the elements of A are all distinct;
- each element of array A is an integer within the range [1..(N + 1)].

### Solution

| | | |
|---|---|---|
| Programming language used: | C# | |
| Total time used: | 5 minutes | ? |
| Effective time used: | 5 minutes | ? |
| Notes: | *not defined yet* | |

### Task timeline ?

09:32:59 09:37:29

Code: 09:37:29 UTC, cs, final, score: **100**

show code in pop-up

```
using System;
// you can also use other imports, for example:
// using System.Collections.Generic;

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

class Solution {
    public int solution(int[] A) {
          int  len = A.Length + 1,
sum = (int)Math.Ceiling(len/2.0)*(len+(len+1)%2),

    for(int i = -1;++i<A.Length;)
        summ+=A[i];

    return sum-summ;
    }
}
```

## Analysis summary

The solution obtained perfect score.

## Analysis

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

expand all **Example tests**

▶ example ✓ OK
example test

expand all **Correctness tests**

▶ empty_and_single ✓ OK
empty list and single element

▶ missing_first_or_last ✓ OK
the first or the last element is missing

▶ single ✓ OK
single element

▶ double ✓ OK
two elements

▶ simple ✓ OK
simple test

expand all **Performance tests**

▶ medium1 ✓ OK
medium test, length = ~10,000

▶ medium2 ✓ OK
medium test, length = ~10,000

▶ large_range ✓ OK
range sequence, length = ~100,000

▶ large1 ✓ OK
large test, length = ~100,000

▶ large2 ✓ OK
large test, length = ~100,000

Codility_

# CodeCheck Report: training6PNYVJ-RJR

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

## Tasks summary

| Task | Time spent | Score |
|---|---|---|
| FrogJmp<br>C# ⚠ | 2 min | 100% |

## Total score

## Tasks Details

Easy

1. FrogJmp

Count minimal number of jumps from position X to Y.

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

### Task description

A small frog wants to get to the other side of the road. The frog is currently located at position X and wants to get to a position greater than or equal to Y. The small frog always jumps a fixed distance, D.

Count the minimal number of jumps that the small frog must perform to reach its target.

Write a function:

```
class Solution { public int solution(int X, int
Y, int D); }
```

that, given three integers X, Y and D, returns the minimal number of jumps from position X to a position equal to or greater than Y.

For example, given:

```
X = 10
Y = 85
D = 30
```

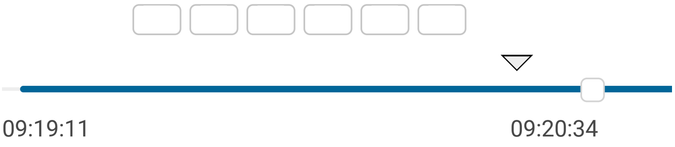
the function should return 3, because the frog will be positioned as follows:

- after the first jump, at position 10 + 30 = 40

### Solution

| | |
|---|---|
| Programming language used: | C# |
| Total time used: | 2 minutes |
| Effective time used: | 2 minutes |
| Notes: | *not defined yet* |

### Task timeline



Code: 09:20:34 UTC, cs, final, score: **100**

show code in pop-up

```
using System;
// you can also use other imports, for example:
// using System.Collections.Generic;
```

- after the second jump, at position 10 + 30 + 30 = 70
- after the third jump, at position 10 + 30 + 30 + 30 = 100

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

- X, Y and D are integers within the range [1..1,000,000,000];
- X ≤ Y.

```

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

class Solution {
    public int solution(int X, int Y, int D) {
        return (int)Math.Ceiling(((double)Y - X)/D
    }
}
```

## Analysis summary

The solution obtained perfect score.

## Analysis

Detected time complexity: O(1)

| | |
|---|---|
| expand all **Example tests** | |
| ▶ example<br>example test | ✓ OK |
| expand all **Correctness tests** | |
| ▶ simple1<br>simple test | ✓ OK |
| ▶ simple2 | ✓ OK |
| ▶ extreme_position<br>no jump needed | ✓ OK |
| ▶ small_extreme_jump<br>one big jump | ✓ OK |
| expand all **Performance tests** | |
| ▶ many_jump1<br>many jumps, D = 2 | ✓ OK |
| ▶ many_jump2<br>many jumps, D = 99 | ✓ OK |
| ▶ many_jump3<br>many jumps, D = 1283 | ✓ OK |
| ▶ big_extreme_jump<br>maximal number of jumps | ✓ OK |
| ▶ small_jumps<br>many small jumps | ✓ OK |

Codility_

## CodeCheck Report: trainingZVE4Y5-NAS

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

### Tasks summary

| Task | | Time spent | Score |
|---|---|---|---|
| OddOccurrencesInArray<br>C# | ⚠ | 8 min | 100% |

### Total score

100%

## Tasks Details

| Easy | 1.<br>OddOccurrencesInArray<br>Find value that occurs in odd number of elements. | Task Score | Correctness | Performance |
|---|---|---|---|---|
| | | 100% | 100% | 100% |

### 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:

### Solution

| | |
|---|---|
| Programming language used: | C# |
| Total time used: | 8 minutes |
| Effective time used: | 8 minutes |
| Notes: | *not defined yet* |

### Task timeline

09:08:31 09:16:08

Code: 09:16:08 UTC, cs, final, score: **100**

show code in pop-up

```
using System;
// you can also use other imports, for example:
using System.Collections.Generic;
```

```
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];
- 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.

```

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

class Solution {
    public int solution(int[] A) {
        // Implement your solution here
        var hash = new HashSet<int>();
        foreach(var i in A)
            if(hash.Contains(i))
                hash.Remove(i);
            else
                hash.Add(i);
        foreach (var i in hash)
            return i;
        return 0;
    }
}
```

## Analysis summary

The solution obtained perfect score.

## Analysis

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

| expand all | Example tests |
|---|---|
| ▶ example1<br>example test | ✓ OK |
| **expand all** | **Correctness tests** |
| ▶ simple1<br>simple test n=5 | ✓ OK |
| ▶ simple2<br>simple test n=11 | ✓ OK |
| ▶ extreme_single_item<br>[42] | ✓ OK |
| ▶ small1<br>small random test n=201 | ✓ OK |
| ▶ small2<br>small random test n=601 | ✓ OK |
| **expand all** | **Performance tests** |
| ▶ medium1<br>medium random test n=2,001 | ✓ OK |
| ▶ medium2<br>medium random test n=100,003 | ✓ OK |
| ▶ big1<br>big random test n=999,999, multiple repetitions | ✓ OK |
| ▶ big2<br>big random test n=999,999 | ✓ OK |

Codility_

## CodeCheck Report: training5PRPKX-QX4

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

### Tasks summary

| Task | Time spent | Score |
|---|---|---|
| CyclicRotation ⚠<br>C# | 7 min | 100% |

### Total score

100%

## Tasks Details

Easy

### 1. CyclicRotation

Rotate an array to the right by a given number of steps.

| Task Score | Correctness | Performance |
|---|---|---|
| 100% | 100% | Not assessed |

### Task description

An array A consisting of N integers is given. Rotation of the array means that each element is shifted right by one index, and the last element of the array is moved to the first place. For example, the rotation of array A = [3, 8, 9, 7, 6] is [6, 3, 8, 9, 7] (elements are shifted right by one index and 6 is moved to the first place).

The goal is to rotate array A K times; that is, each element of A will be shifted to the right K times.

Write a function:

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

that, given an array A consisting of N integers and an integer K, returns the array A rotated K times.

For example, given

```
A = [3, 8, 9, 7, 6]
K = 3
```

the function should return [9, 7, 6, 3, 8]. Three rotations were made:

```
[3, 8, 9, 7, 6] -> [6, 3, 8, 9, 7]
[6, 3, 8, 9, 7] -> [7, 6, 3, 8, 9]
```

### Solution

| | |
|---|---|
| Programming language used: | C# |
| Total time used: | 7 minutes |
| Effective time used: | 7 minutes |
| Notes: | *not defined yet* |

### Task timeline

09:00:01 09:06:12

Code: 09:06:11 UTC, cs, final, score: **100**

show code in pop-up

```
using System;
// you can also use other imports, for example:
// using System.Collections.Generic;
```

```
[7, 6, 3, 8, 9] -> [9, 7, 6, 3, 8]
```

For another example, given

```
A = [0, 0, 0]
K = 1
```

the function should return [0, 0, 0]

Given

```
A = [1, 2, 3, 4]
K = 4
```

the function should return [1, 2, 3, 4]

Assume that:

- N and K are integers within the range [0..100];
- each element of array A is an integer within the range [−1,000..1,000].

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

```

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

class Solution {
    public int[] solution(int[] A, int K) {
        var rotArr = new int[A.Length];
        if(rotArr.Length > 0)
            for (int i = -1, j = A.Length - (K%A.L
            rotArr[i] = A[(j+i)% A.Length];
        return rotArr;
    }
}
```

## Analysis summary

The solution obtained perfect score.

## Analysis

expand all **Example tests**

▶ example ✓ OK
first example test

▶ example2 ✓ OK
second example test

▶ example3 ✓ OK
third example test

expand all **Correctness tests**

▶ extreme_empty ✓ OK
empty array

▶ single ✓ OK
one element, 0 <= K <= 5

▶ double ✓ OK
two elements, K <= N

▶ small1 ✓ OK
small functional tests, K < N

▶ small2 ✓ OK
small functional tests, K >= N

▶ small_random_all_rotations ✓ OK
small random sequence, all rotations, N = 15

▶ medium_random ✓ OK
medium random sequence, N = 100

▶ maximal ✓ OK
maximal N and K

Codility_

# CodeCheck Report: trainingWN9JN2-DDY

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

## Tasks summary

| Task | Time spent | Score |
|---|---|---|
| BinaryGap ⚠<br>C# | 10 min | 100% |

## Total score

## Tasks Details

### 1. BinaryGap

Easy

Find longest sequence of zeros in binary representation of an integer.

| Task Score | Correctness | Performance |
|---|---|---|
| 100% | 100% | Not assessed |

### Task description

A *binary gap* within a positive integer N is any maximal sequence of consecutive zeros that is surrounded by ones at both ends in the binary representation of N.

For example, number 9 has binary representation `1001` and contains a binary gap of length 2. The number 529 has binary representation `1000010001` and contains two binary gaps: one of length 4 and one of length 3. The number 20 has binary representation `10100` and contains one binary gap of length 1. The number 15 has binary representation `1111` and has no binary gaps. The number 32 has binary representation `100000` and has no binary gaps.

Write a function:

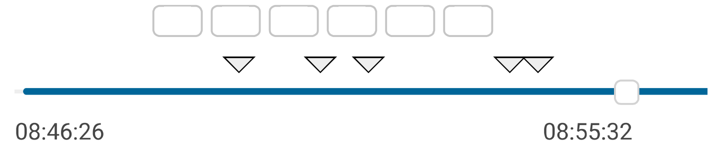
```
class Solution { public int solution(int N); }
```

that, given a positive integer N, returns the length of its longest binary gap. The function should return 0 if N doesn't contain a

### Solution

| | |
|---|---|
| Programming language used: | C# |
| Total time used: | 10 minutes |
| Effective time used: | 10 minutes |
| Notes: | *not defined yet* |

### Task timeline



binary gap.

For example, given N = 1041 the function should return 5, because N has binary representation `10000010001` and so its longest binary gap is of length 5. Given N = 32 the function should return 0, because N has binary representation '100000' and thus no binary gaps.

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

- N is an integer within the range [1..2,147,483,647].

Code: 08:55:32 UTC, cs, final, score: **100**

show code in pop-up

```
using System;
// you can also use other imports, for example:
// using System.Collections.Generic;

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

class Solution {
        public int solution(int N) {
                bool a = false;
                int len = 0, max = 0;
                for (uint i = 1; i <= N; i <<= 1) {
                        if ((i & N) != 0) {
                                if (a) {
                                        if (len > m
                                                max
                                        len = 0;
                                }
                                a = true;
                        }
                        else if (a)
                                ++len;
                }
                return max;
        }
}
```

## Analysis summary

The solution obtained perfect score.

## Analysis

expand all **Example tests**

▶ example1 ✓ OK
example test n=1041=10000010001_2

▶ example2 ✓ OK
example test n=15=1111_2

▶ example3 ✓ OK
example test n=32=100000_2

expand all **Correctness tests**

▶ extremes ✓ OK
n=1, n=5=101_2 and n=2147483647=2**31-1

▶ trailing_zeroes ✓ OK
n=6=110_2 and n=328=101001000_2

▶ power_of_2 ✓ OK
n=5=101_2, n=16=2**4 and n=1024=2**10

▶ simple1 ✓ OK
n=9=1001_2 and n=11=1011_2

▶ simple2 ✓ OK
n=19=10011 and n=42=101010_2

| | | |
|---|---|---|
| ▶ | simple3<br>n=1162=10010001010_2 and n=5=101_2 | ✓ OK |
| ▶ | medium1<br>n=51712=1100101000000000_2 and n=20=10100_2 | ✓ OK |
| ▶ | medium2<br>n=561892=10001001001011100100_2 and n=9=1001_2 | ✓ OK |
| ▶ | medium3<br>n=66561=10000010000000001_2 | ✓ OK |
| ▶ | large1<br>n=6291457=11000000000000000000001_2 | ✓ OK |
| ▶ | large2<br>n=74901729=100011101101110100011100001 | ✓ OK |
| ▶ | large3<br>n=805306373=110000000000000000000000000101_2 | ✓ OK |
| ▶ | large4<br>n=1376796946=1010010000100000100000100010010_2 | ✓ OK |
| ▶ | large5<br>n=1073741825=1000000000000000000000000000001_2 | ✓ OK |
| ▶ | large6<br>n=1610612737=1100000000000000000000000000001_2 | ✓ OK |