DAY 2

CODILITY LINKS AND SCREENSHOTS

L1:

Binary Gap - https://app.codility.com/demo/results/trainingPAU3Z2-H35/

L2:

Cyclic Rotation - https://app.codility.com/demo/results/trainingS9JJSJ-8ZH/

Odd Occurence in Array - https://app.codility.com/demo/results/training3GTPKM-ZXZ/

L3:

Frog Jump - https://app.codility.com/demo/results/trainingVZFKQ8-6MX/

Perm Missing Element - https://app.codility.com/demo/results/trainingC9M2JV-DGB/

Tape Equilibrium - https://app.codility.com/demo/results/trainingDFG7BC-GN5/

FURY ROAD:

https://app.codility.com/demo/results/training7UM265-628/

Codility_

CodeCheck Report: trainingPAU3Z2-H35

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
BinaryGap ⚠ C++	10 min	100%

Total score

100%

Tasks Details

Easy

1. **BinaryGap**
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:

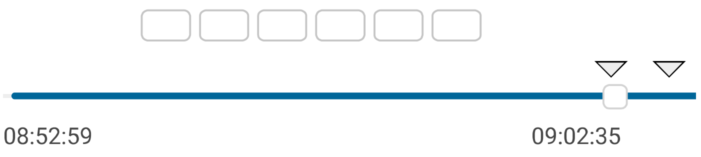
```
int solution(int N);
```

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].

Copyright 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 09:02:35 UTC, cpp, final, score: **100** show code in pop-up

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

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

int solution(int N) {
    int numOfBits = floor(std::log2(N))+1;
    bool ifOne = false;
    int curGap = 0;
    int maxGap = 0;

    for(int i=0; i<numOfBits; i++){
        if(ifOne && !(N &(1<<i))){
            curGap++;
        }
        else if(N & (1<<i)){
            if(ifOne){
                if(curGap>maxGap){
                    maxGap = curGap;
                }
                curGap = 0;
            }
            ifOne = true;
        }
    }
    return maxGap;
}
```

Analysis summary

The solution obtained perfect score.

Analysis

expand all	Example tests	
▶	example1 example test n=1041=10000010001_2	✓ OK
▶	example2 example test n=15=1111_2	✓ OK
▶	example3 example test n=32=100000_2	✓ OK
expand all	Correctness tests	
▶	extremes n=1, n=5=101_2 and n=2147483647=2**31-1	✓ OK
▶	trailing_zeroes n=6=110_2 and n=328=101001000_2	✓ OK
▶	power_of_2 n=5=101_2, n=16=2**4 and n=1024=2**10	✓ OK
▶	simple1 n=9=1001_2 and n=11=1011_2	✓ OK

▶ simple2 n=19=10011 and n=42=101010_2	✓ OK
▶ simple3 n=1162=10010001010_2 and n=5=101_2	✓ OK
▶ medium1 n=51712=110010100000000_2 and n=20=10100_2	✓ OK
▶ medium2 n=561892=10001001001011100100_2 and n=9=1001_2	✓ OK
▶ medium3 n=66561=10000010000000001_2	✓ OK
▶ large1 n=6291457=11000000000000000000001_2	✓ OK
▶ large2 n=74901729=1000111011011101000111 00001	✓ OK
▶ large3 n=805306373=110000000000000000000000000101_2	✓ OK
▶ large4 n=1376796946=1010010000100000100000100010010_2	✓ OK
▶ large5 n=1073741825=1000000000000000000000000000001_2	✓ OK
▶ large6 n=1610612737=1100000000000000000000000000001_2	✓ OK

Codility_

CodeCheck Report: trainingS9JJSJ-8ZH

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
CyclicRotation C++ ⚠	11 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:

```
vector<int> solution(vector<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
```

Solution

Programming language used:	C++
Total time used:	11 minutes
Effective time used:	11 minutes
Notes:	*not defined yet*

Task timeline

09:15:21 09:25:28

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]
[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.

Copyright 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 09:25:28 UTC, cpp, final, score: **100**   show code in pop-up

```cpp
// you can use includes, for example:
// #include <algorithm>
#include <vector>
// you can write to stdout for debugging purposes,
// cout << "this is a debug message" << endl;

vector<int> solution(vector<int> &A, int K) {
    int n = A.size();
    vector<int> result(n,0);

    if(!n){
        return result;
    }

    int startFrom = 0;
    if(K%n){
        startFrom = n-K%n;
    }

    for(int i = 0; i<n; i++){
        result[i] = A[startFrom];
        startFrom = ((startFrom+1)%n);
    }

    return result;
}
```

Analysis summary

The solution obtained perfect score.

Analysis

expand all	Example tests	
▶	example first example test	✓ OK
▶	example2 second example test	✓ OK
▶	example3 third example test	✓ OK
expand all	Correctness tests	
▶	extreme_empty empty array	✓ OK
▶	single one element, 0 <= K <= 5	✓ OK
▶	double two elements, K <= N	✓ OK
▶	small1 small functional tests, K < N	✓ OK
▶	small2 small functional tests, K >= N	✓ OK
▶	small_random_all_rotations small random sequence, all rotations, N = 15	✓ OK

▶	medium_random medium random sequence, N = 100	✓ OK
▶	maximal maximal N and K	✓ OK

Codility_

CodeCheck Report: training3GTPKM-ZXZ

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
OddOccurrencesInArray C++ ⚠	7 min	100%

Total score

100%

Tasks Details

Easy

1. **OddOccurrencesInArray**
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.

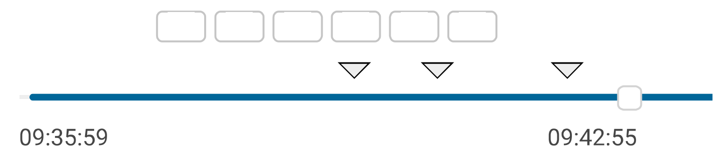
Write a function:

```
int solution(vector<int> &A);
```

Solution

Programming language used:	C++
Total time used:	7 minutes
Effective time used:	7 minutes
Notes:	*not defined yet*

Task timeline



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];
- 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.

Copyright 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 09:42:54 UTC, cpp, final, score: **100** show code in pop-up

```cpp
// you can use includes, for example:
// #include <algorithm>
#include<vector>
#include<unordered_map>
// you can write to stdout for debugging purposes,
// cout << "this is a debug message" << endl;

int solution(vector<int> &A) {
    unordered_map<int,int> counts;

    for(int n : A){
        counts[n]++;
    }

    for(auto const& pair : counts){
        if(pair.second%2 != 0){
            return pair.first;
        }
    }

    return -1;
}
```

Analysis summary

The solution obtained perfect score.

Analysis

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

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	✓ OK
▶ big1 big random test n=999,999, multiple repetitions	✓ OK
▶ big2 big random test n=999,999	✓ OK

Codility_

CodeCheck Report: trainingVZFKQ8-6MX

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
FrogJmp ⚠ C++	7 min	100%

Total score

100%

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:

```
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
```

Solution

Programming language used:	C++
Total time used:	7 minutes
Effective time used:	7 minutes
Notes:	*not defined yet*

Task timeline

09:50:19 09:56:37

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

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

Copyright 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 09:56:37 UTC, cpp, final, score: **100**

show code in pop-up

```cpp
// you can use includes, for example:
// #include <algorithm>
#include<cmath>
// you can write to stdout for debugging purposes,
// cout << "this is a debug message" << endl;

int solution(int X, int Y, int D) {
    double ans,distToBeCovered;

    distToBeCovered = (double)Y-(double)X;

    ans = ceil(distToBeCovered / ((double)D));

    return ans;
}
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(1)**

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

Codility_

Check out Codility training tasks

Tasks Details

		Task Score	Correctness	Performance
Easy	1. PermMissingElem Find the missing element in a given permutation.	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:

```
int solution(vector<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)].

Copyright 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Solution

Programming language used:	C++
Total time used:	2 minutes
Effective time used:	2 minutes
Notes:	*not defined yet*

Task timeline

10:06:02 10:07:59

Code: 10:07:59 UTC, cpp, final, score: **100**

show code in pop-up

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

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

int solution(vector<int> &A) {
    unsigned long long int n = A.size();
    unsigned long long int sum = 0;

    for(unsigned int i = 0; i<n; i++){
        sum+=A[i];
    }

    double result = (n+1)*(n+2)/2 - sum;

    return result;
}
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests	
▶	example example test	✓ OK
expand all	Correctness tests	
▶	empty_and_single empty list and single element	✓ OK
▶	missing_first_or_last the first or the last element is missing	✓ OK
▶	single single element	✓ OK
▶	double two elements	✓ OK
▶	simple simple test	✓ OK
expand all	Performance tests	
▶	medium1 medium test, length = ~10,000	✓ OK
▶	medium2 medium test, length = ~10,000	✓ OK
▶	large_range range sequence, length = ~100,000	✓ OK
▶	large1 large test, length = ~100,000	✓ OK
▶	large2 large test, length = ~100,000	✓ OK

Codility_

CodeCheck Report: trainingDFG7BC-GN5

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
TapeEquilibrium ⚠ C++	7 min	100%

Total score

100%

Tasks Details

Easy

1. **TapeEquilibrium**
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:

Solution

Programming language used:	C++
Total time used:	7 minutes
Effective time used:	7 minutes
Notes:	*not defined yet*

Task timeline

10:14:58 10:21:25

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

Write a function:

```
int solution(vector<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].

Copyright 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 10:21:25 UTC, cpp, final, score: **100**   show code in pop-up

```
// you can use includes, for example:
// #include <algorithm>
#include<cmath>
#include<limits.h>
// you can write to stdout for debugging purposes,
// cout << "this is a debug message" << endl;

int solution(vector<int> &A) {
    int totalSum = 0,leftSum = 0, rightSum = 0;
    int n = A.size();
    int minDiff = INT_MAX;
    for(int a : A){
        totalSum += a;
    }
    for(int i=0; i<(n-1); i++){
        leftSum += A[i];
        rightSum = totalSum - leftSum;

        int diff = abs(leftSum - rightSum);
        if(diff < minDiff){
            minDiff = diff;
        }
    }

    return minDiff;
}
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all **Example tests**

▶ example example test	✓ OK

expand all **Correctness tests**

▶ double two elements	✓ OK
▶ simple_positive simple test with positive numbers, length = 5	✓ OK
▶ simple_negative simple test with negative numbers, length = 5	✓ OK
▶ simple_boundary only one element on one of the sides	✓ OK
▶ small_random random small, length = 100	✓ OK
▶ small_range range sequence, length = ~1,000	✓ OK

▶	small small elements	✓ OK

expand all **Performance tests**

▶	medium_random1 random medium, numbers from 0 to 100, length = ~10,000	✓ OK
▶	medium_random2 random medium, numbers from -1,000 to 50, length = ~10,000	✓ OK
▶	large_ones large sequence, numbers from -1 to 1, length = ~100,000	✓ OK
▶	large_random random large, length = ~100,000	✓ OK
▶	large_sequence large sequence, length = ~100,000	✓ OK
▶	large_extreme large test with maximal and minimal values, length = ~100,000	✓ OK

Codility_

CodeCheck Report: training7UM265-628

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
ScooterRoad C++	5 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?

Solution

Programming language used:	C++
Total time used:	5 minutes
Effective time used:	5 minutes
Notes:	*not defined yet*

Task timeline

11:17:31 11:21:56

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.

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'.

Copyright 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Code: 11:21:56 UTC, cpp, final, score: **100**

show code in pop-up

```cpp
// you can use includes, for example:
// #include <algorithm>
#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] == '
    }

    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 First example test.	✓ OK
▶	example2 Second example test.	✓ OK
▶	example3 Third example test.	✓ OK
expand all	Correctness tests	
▶	very_short_road N = 1.	✓ OK
▶	short_road N <= 3.	✓ OK
▶	only_scooter Only scooter is used.	✓ OK
▶		

	only_walking Scooter is not used at all.	✓ OK
▶	all_asphalt_first Road can be described by "AA...ASS...S". N <= 200.	✓ OK
▶	small_random_change_point Getting off the scooter at a random position. N <= 200.	✓ OK

expand all **Performance tests**

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