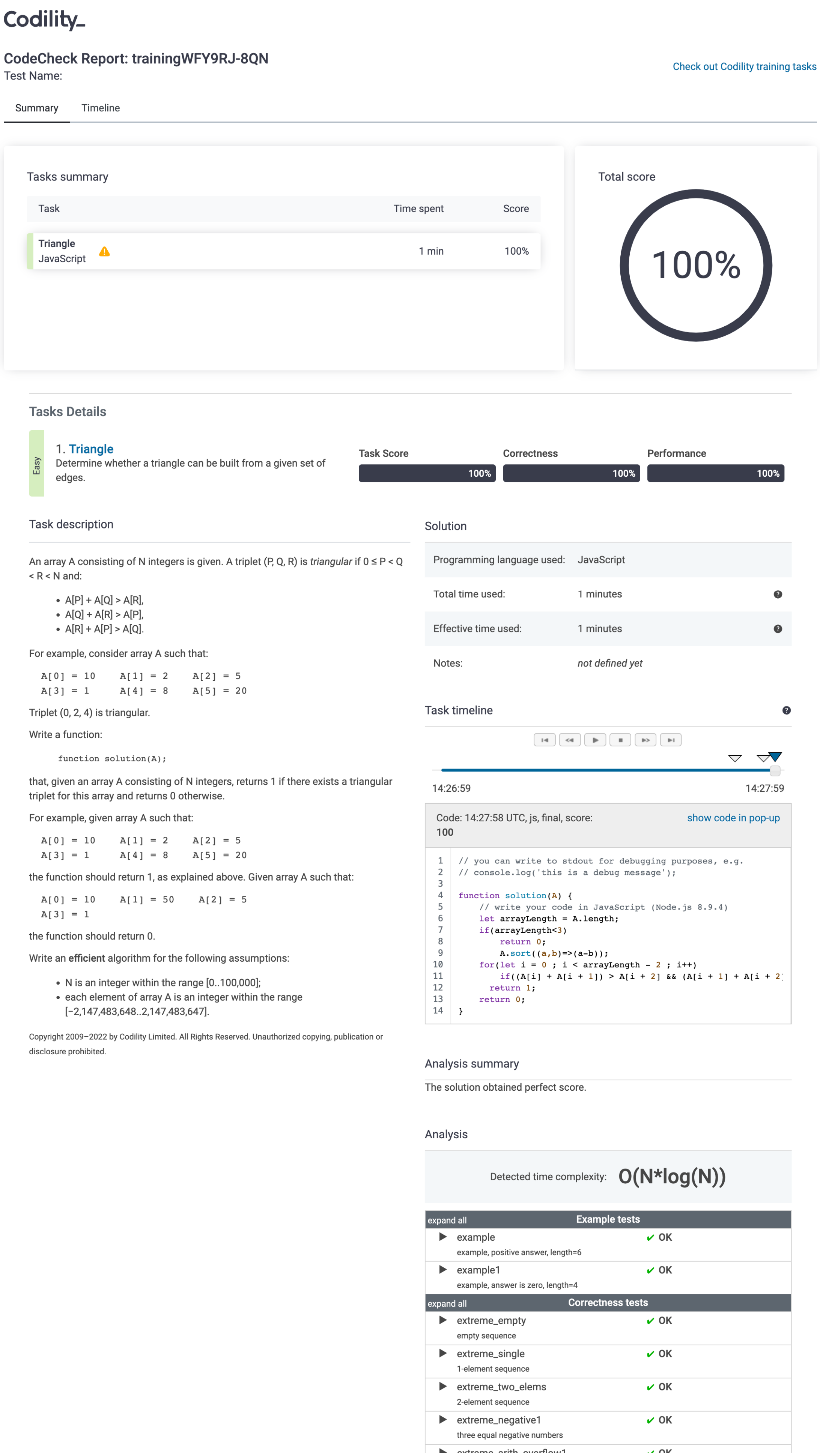


Codility_

CodeCheck Report: trainingWFY9RJ-8QN

Test Name:

Tasks summary

Task	Time spent	Score
Triangle JavaScript ⚠	1 min	100%

Total score

100%

Tasks Details

Easy

1. **Triangle**

Determine whether a triangle can be built from a given set of edges.

Task Score	Correctness	Performance
100%	100%	100%

Task description

An array A consisting of N integers is given. A triplet (P, Q, R) is *triangular* if 0 ≤ P < Q < R < N and:

- A[P] + A[Q] > A[R],
- A[Q] + A[R] > A[P],
- A[R] + A[P] > A[Q].

For example, consider array A such that:

```
A[0] = 10    A[1] = 2    A[2] = 5
A[3] = 1     A[4] = 8    A[5] = 20
```

Triplet (0, 2, 4) is triangular.

Write a function:

```
    function solution(A);
```

that, given an array A consisting of N integers, returns 1 if there exists a triangular triplet for this array and returns 0 otherwise.

For example, given array A such that:

```
A[0] = 10    A[1] = 2    A[2] = 5
A[3] = 1     A[4] = 8    A[5] = 20
```

the function should return 1, as explained above. Given array A such that:

```
A[0] = 10    A[1] = 50    A[2] = 5
A[3] = 1
```

the function should return 0.

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

- N is an integer within the range [0..100,000];
- each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].

Solution

Programming language used:	JavaScript
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline

14:26:59 14:27:59

Code: 14:27:58 UTC, js, final, score: **100**

```
// you can write to stdout for debugging purposes, e.g.
// console.log('this is a debug message');

function solution(A) {
    // write your code in JavaScript (Node.js 8.9.4)
    let arrayLength = A.length;
    if(arrayLength<3)
        return 0;
        A.sort((a,b)=>(a-b));
    for(let i = 0 ; i < arrayLength - 2 ; i++)
        if((A[i] + A[i + 1]) > A[i + 2] && (A[i + 1] + A[i + 2]
      return 1;
    return 0;
}
```

Analysis summary

The solution obtained perfect score.

Analysis

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

Example tests

example ✔ OK
example, positive answer, length=6

example1 ✔ OK
example, answer is zero, length=4

Correctness tests

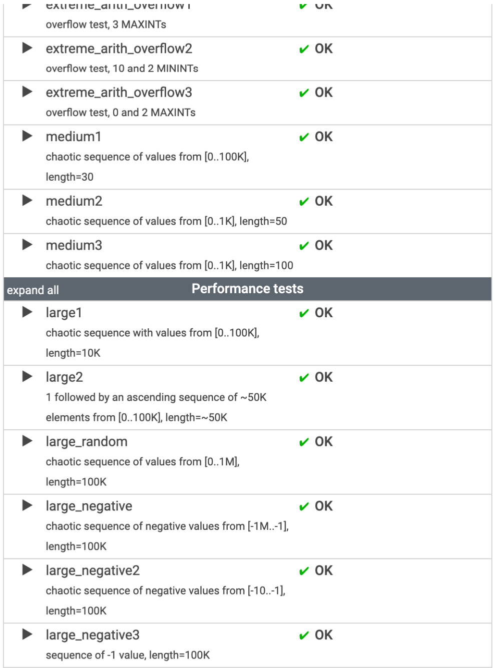
extreme_empty ✔ OK
empty sequence

extreme_single ✔ OK
1-element sequence

extreme_two_elems ✔ OK
2-element sequence

extreme_negative1 ✔ OK
three equal negative numbers

extreme_arith_overflow1 ✔ OK



▶ extreme_arith_overflow1 overflow test, 3 MAXINTs	✔ OK
▶ extreme_arith_overflow2 overflow test, 10 and 2 MININTs	✔ OK
▶ extreme_arith_overflow3 overflow test, 0 and 2 MAXINTs	✔ OK
▶ medium1 chaotic sequence of values from [0..100K], length=30	✔ OK
▶ medium2 chaotic sequence of values from [0..1K], length=50	✔ OK
▶ medium3 chaotic sequence of values from [0..1K], length=100	✔ OK
expand all **Performance tests**	
▶ large1 chaotic sequence with values from [0..100K], length=10K	✔ OK
▶ large2 1 followed by an ascending sequence of ~50K elements from [0..100K], length=~50K	✔ OK
▶ large_random chaotic sequence of values from [0..1M], length=100K	✔ OK
▶ large_negative chaotic sequence of negative values from [-1M..-1], length=100K	✔ OK
▶ large_negative2 chaotic sequence of negative values from [-10..-1], length=100K	✔ OK
▶ large_negative3 sequence of -1 value, length=100K	✔ OK