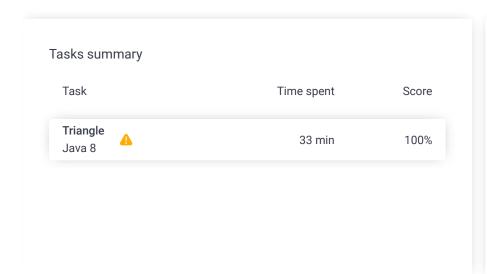
Codility_

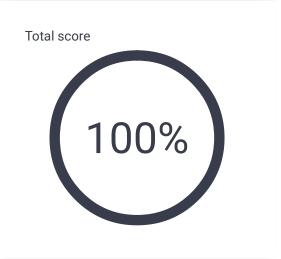
CodeCheck Report: trainingZS5Y5S-SV3

Test Name:

Summary Timeline

Check out Codility training tasks





Tasks Details

1. Triangle

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

Task Score

100%

Correctness

Performance

100%

Task description

An array A consisting of N integers is given. A triplet (P, Q, R) is triangular if $0 \le P < Q < R < N$ 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:

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:

Solution

Programming language used: Java 8

Total time used: 33 minutes

100%

Effective time used: 33 minutes

Notes: not defined yet

Task timeline

02:46:07 03:19:00

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

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].

Copyright 2009–2021 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Test results - Codility

Code: 03:18:59 UTC, java,

```
final, score: 100
 1
     // you can also use imports, for example:
 2
     // import java.util.*;
 3
 4
     // you can write to stdout for debugging purposes,
 5
     // System.out.println("this is a debug message");
 6
7
     class Solution {
 8
          public int solution(int[] A) {
 9
              final int A_LENGTH = A.length;
10
              if (A_LENGTH < 3) {</pre>
11
12
                  return 0;
13
              }
14
15
              quicksort(A, 0, A_LENGTH - 1);
16
17
              for (int i = 2; i < A_LENGTH; i++) {</pre>
18
                  if ((long)A[i] < (long)A[i - 2] + (long)
19
                       return 1;
20
                  }
21
              }
22
23
              return 0;
24
          }
25
26
          private void quicksort(int arr[], int left, int
27
              int index = partition(arr, left, right);
28
29
              if (left < index - 1) {</pre>
30
                  quicksort(arr, left, index - 1);
31
32
33
              if (index < right) {</pre>
34
                  quicksort(arr, index, right);
35
              }
36
          }
37
38
          private int partition(int arr[], int left, int
39
              int pivot = arr[(left + right) / 2];
40
41
              while (left <= right) {
                  while (arr[left] < pivot) {</pre>
42
43
                       left++;
44
                  }
45
46
                  while (arr[right] > pivot) {
47
                       right--;
48
49
50
                  if (left <= right) {</pre>
51
                       int tmp = arr[left];
52
                       arr[left++] = arr[right];
53
                       arr[right--] = tmp;
54
                  }
55
              }
56
57
              return left;
58
          }
59
     }
```

show code in pop-up

Analysis summary

The solution obtained perfect score.

Analysis

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

-	nd all Example tes	
•	example example, positive answer, length=6	✓ OK
•	example1 example, answer is zero, length=4	✓ OK
expand all Correctness tests		
•	extreme_empty empty sequence	✓ OK
•	extreme_single 1-element sequence	✓ OK
•	extreme_two_elems 2-element sequence	✓ OK
•	extreme_negative1 three equal negative numbers	✓ 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 [0100K], length=30	√ OK
•	medium2 chaotic sequence of values from [01K], length=50	√ OK
•	medium3 chaotic sequence of values from [01K], length=100	√ OK
expand all Performance tests		
•	large1 chaotic sequence with values from [0100K], length=10K	√ OK
•	large2 1 followed by an ascending sequence of ~50K elements from [0100K], length=~50K	✓ OK
•	large_random chaotic sequence of values from [01M], length=100K	√ OK
•	large_negative chaotic sequence of negative values from [-1M1], length=100K	✓ OK
•	large_negative2 chaotic sequence of negative values from [-101], length=100K	√ OK
•	large_negative3 sequence of -1 value, length=100K	✓ OK