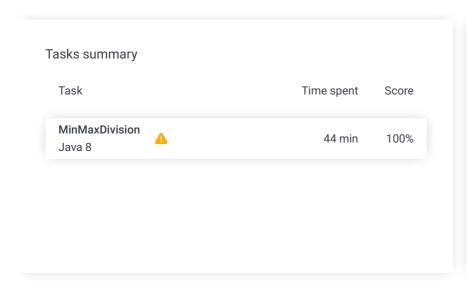
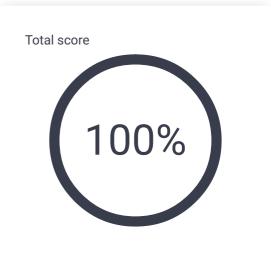
Codility_

CodeCheck Report: trainingFH9G9N-5U9

Test Name:

Summary Timeline Check out Codility training tasks





Tasks Details

1. MinMaxDivision

Divide array A into K blocks and minimize the largest sum of any block.

Task Score

Correctness

100%

Performance

100%

100%

Task description

You are given integers K, M and a non-empty array A consisting of N integers. Every element of the array is not greater than M.

You should divide this array into K blocks of consecutive elements. The size of the block is any integer between 0 and N. Every element of the array should belong to some block.

The sum of the block from X to Y equals A[X] + A[X + 1] + ... +A[Y]. The sum of empty block equals 0.

The large sum is the maximal sum of any block.

For example, you are given integers K = 3, M = 5 and array A such that:

- A[0] = 2
- A[1] = 1
- A[2] = 5
- A[3] = 1
- A[4] = 2
- A[5] = 2
- A[6] = 2

The array can be divided, for example, into the following blocks:

• [2, 1, 5, 1, 2, 2, 2], [], [] with a large sum of 15;

Solution

Programming language used: Java 8

Total time used: 44 minutes

Effective time used: 44 minutes

Notes: not defined yet

Task timeline

21:30:30 22:13:58

Code: 22:13:58 UTC, java, show code in pop-up final, score: 100

// you can also use imports, for example: 1 2 // import java.util.*;

3

- [2], [1, 5, 1, 2], [2, 2] with a large sum of 9;
- [2, 1, 5], [], [1, 2, 2, 2] with a large sum of 8;
- [2, 1], [5, 1], [2, 2, 2] with a large sum of 6.

The goal is to minimize the large sum. In the above example, 6 is the minimal large sum.

Write a function:

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

that, given integers K, M and a non-empty array A consisting of N integers, returns the minimal large sum.

For example, given K = 3, M = 5 and array A such that:

A[0] = 2 A[1] = 1 A[2] = 5 A[3] = 1 A[4] = 2 A[5] = 2

the function should return 6, as explained above.

Write an efficient algorithm for the following assumptions:

- N and K are integers within the range [1..100,000];
- M is an integer within the range [0..10,000];
- each element of array A is an integer within the range [0..M].

Copyright 2009–2023 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
4
     // you can write to stdout for debugging purpo
     // System.out.println("this is a debug message
 6
 7
     class Solution {
         public int solution(int K, int M, int[] A)
 8
 9
             int N = A.length;
10
             int min = 0;
11
12
             int[] prefixSum = new int[N + 1];
             for (int i = 0; i < N; i++) {
13
14
                  prefixSum[i + 1] = prefixSum[i] +
15
                  min = Math.max(min, A[i]);
16
17
             int max = Math.min(prefixSum[N], (N +
18
19
             min = Math.max(min, (max + K - 1) / K)
20
21
             while (max > min) {
22
                  int mid = (min + max) / 2;
23
                  if (validateDivision(mid, K, prefi
24
                      max = mid:
25
                  } else {
26
                      min = mid + 1;
27
28
             }
29
30
             return min;
31
32
         private boolean validateDivision(int group
33
34
             int count = 0;
35
             int prefixSumLimit = 0;
36
37
             for (int i = 1; i < prefixSum.length;</pre>
38
                  if (prefixSum[i] > prefixSumLimit)
39
                      count++:
40
                      if (count > K) {
41
                          return false;
42
43
44
                      prefixSumLimit = prefixSum[i -
45
                  }
46
             }
47
48
             return true;
49
         }
50
     }
```

Analysis summary

The solution obtained perfect score.

Analysis

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

```
Example tests
expand all
                                   ✓ OK
   example
    example test
                    Correctness tests
expand all
extreme_single
                                   ✓ OK
    single elements
extreme_double
                                   ✓ OK
    single and double elements
    extreme_min_max
                                   ✓ OK
    maximal / minimal values
```

•	simple1 simple tests	∠ OK
•	simple2 simple tests	✓ OK
•	tiny_random_ones random values {0, 1}, N = 100	✓ OK
expar	nd all Performance to	ests
•	small_random_ones random values {0, 1}, N = 100	✓ OK
•	medium_zeros many zeros and 99 in the middle, length = 15,000	∨ OK
•	medium_random random values {1, 100}, N = 20,000	∨ OK
•	large_random random values {0,, MAX_INT}, N = 100,000	∨ OK
•	large_random_ones random values {0, 1}, N = 100,000	∨ OK
•	all_the_same all the same values, N = 100,000	∨ OK