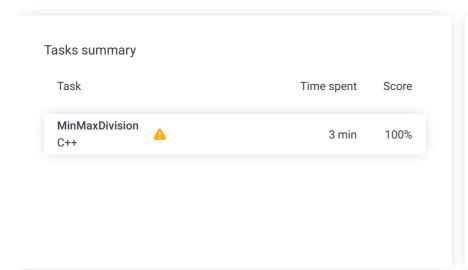
Codility_

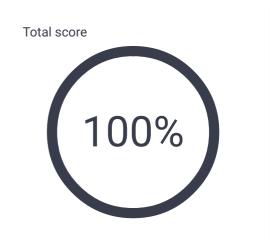
CodeCheck Report: trainingBYV735-59J

Test Name:

Check out Codility training tasks

Al Assistant Transcript Summary Timeline





Tasks Details

1. MinMaxDivision

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

100%

Correctness

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:

3 minutes Total time used:

Effective time used: 3 minutes

Notes: not defined yet

Task timeline

12:03:20 12:05:33

Code: 12:05:33 UTC, cpp, final, score: 100

show code in pop-up

- 1 // you can use includes, for example:
- 2 #include <algorithm>
- 3 #include <numeric>

a

- [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:

```
int solution(int K, int M, vector<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
- A[6] = 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 2009–2024 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
Test results - Codility
```

```
// you can write to stdout for debugging purpo
6
     // cout << "this is a debug message" << endl;</pre>
    bool division(vector<int> &A, int max, int K)
8
9
         int cur = 0;
10
         int count = 1;
11
         for (int a : A) {
12
13
             if (cur + a > max) {
14
                 cur = a:
15
                 count++;
16
                 if (count > K) {
17
                     return false;
18
                 }
19
             } else {
20
                 cur += a;
21
22
         }
23
         return true;
24
     }
25
26
     int solution(int K, int M, vector<int> &A) {
27
         int lower = *max_element(A.begin(), A.end
28
         int upper = accumulate(A.begin(), A.end(),
29
30
         int result = upper;
31
         while (lower <= upper) {
32
33
             int mid = lower + (upper - lower) / 2;
34
             if (division(A, mid, K)) { //합이 mid보
                 result = mid; //k개로 나누기 성공했으니
35
                 upper = mid - 1; //이진탐색
36
37
             } else { //k개보다 많게 나눠야하면
38
                 lower = mid + 1; //이진탐색
39
40
         }
41
42
         return result;
43
    }
```

Analysis summary

The solution obtained perfect score.

Analysis

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

xpand all	Exampl	e tests
example example test	Елипрі	✓ OK
xpand all	Correctne	ess tests
extreme_s single element	9	∨ OK
extreme_d single and do	ouble uble elements	∨ OK
extreme_m maximal / mi	_	∨ OK
simple1		∨ OK
simple2		✓ OK
tiny_rando	m_ones es {0, 1}, N = 100	∨ OK

Test results - Codility

expand all	Performance tes	ts
► small_random, random values {0,		∕ OK
medium_zeros many zeros and 96 length = 15,000		⁄ OK
► medium_rando random values {1,		∕ OK
large_random random values {0, 100,000		⁄ OK
► large_random_ random values {0,		∕ OK
► all_the_same all the same value	_	∕ OK