Codility_

CodeCheck Report: trainingNXXRN6-5QR

Test Name:

Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
MinMaxDivision C++	37 min	50%

Total score

Tasks Details

Medium

1. MinMaxDivision

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

Task Score	Correctness	Performance
50%	50%	50%

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
```

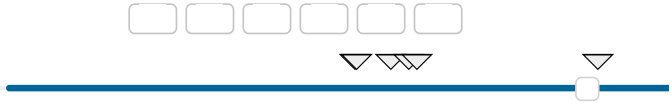
The array can be divided, for example, into the following blocks:

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

Solution

Programming language used:	C++
Total time used:	37 minutes
Effective time used:	37 minutes
Notes:	*not defined yet*

Task timeline



11:24:25 12:01:20

Code: 12:01:16 UTC, cpp, final, score: **50** show code in pop-up

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

- [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].


```
4
5   // you can write to stdout for debugging purpo
6   // cout << "this is a debug message" << endl;
7
8   bool division(vector<int> &A, int max, int K)
9       int cur = 0;
10      int count = 1;
11
12      for (int a : A) {
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      //M이 최대값
28
29      //K개의 블록으로 배열 자르기 (k=3이어도 [],[],[0:M
30      //블록의 인자들을 다 더했을 때의 값이 가장 작은 경우를
31      //그 작은 합을 반환
32
33      //아무튼 k개만큼 자르는게 좋음 (빈배열x)
34
35      int lower = M;
36      //*max_element(A.begin(), A.end()); //나눠
37      int upper = accumulate(A.begin(), A.end(),
38
39      int result = upper;
40
41      while (lower <= upper) {
42          int mid = lower + (upper - lower) / 2;
43          if (division(A, mid, K)) { //합이 mid보
44              result = mid; //k개로 나누기 성공했으니
45              upper = mid - 1; //이진탐색
46          } else { //k개보다 많게 나눠야하면
47              lower = mid + 1; //이진탐색
48          }
49      }
50
51      return result;
52  }
```

Analysis summary

The following issues have been detected: wrong answers.

For example, for the input `(2, 10, [4, 4])` the solution returned a wrong answer (got 8 expected 4).

Analysis

expand all	Example tests	
▶	example example test	✔ OK
expand all	Correctness tests	
▶	extreme_single single elements	✔ OK
▶	extreme_double single and double elements	✘ WRONG ANSWER got 8 expected 4
▶	extreme_min_max maximal / minimal values	✔ OK

▶	simple1 simple tests	✔ OK
▶	simple2 simple tests	✘ WRONG ANSWER got 1000 expected 999
▶	tiny_random_ones random values {0, 1}, N = 100	✘ WRONG ANSWER got 8 expected 3

expand all **Performance tests**

▶	small_random_ones random values {0, 1}, N = 100	✘ WRONG ANSWER got 53 expected 27
▶	medium_zeros many zeros and 99 in the middle, length = 15,000	✘ WRONG ANSWER got 100 expected 99
▶	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	✘ WRONG ANSWER got 10000 expected 452
▶	all_the_same all the same values, N = 100,000	✔ OK