Codility_

CodeCheck Report: trainingJWMAXE-52H

Test Name:

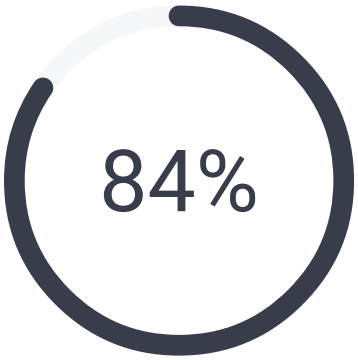
Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxSliceSum C++ ⚠	2 min	84%

Total score



Tasks Details

Easy

1. MaxSliceSum

Find a maximum sum of a compact subsequence of array elements.

Task Score	Correctness	Performance
84%	100%	60%

Task description

A non-empty array A consisting of N integers is given. A pair of integers (P, Q), such that 0 ≤ P ≤ Q < N, is called a *slice* of array A. The *sum* of a slice (P, Q) is the total of A[P] + A[P+1] + ... + A[Q].

Write a function:

```
int solution(vector<int> &A);
```

that, given an array A consisting of N integers, returns the maximum sum of any slice of A.

For example, given array A such that:

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

the function should return 5 because:

- (3, 4) is a slice of A that has sum 4,
- (2, 2) is a slice of A that has sum −6,
- (0, 1) is a slice of A that has sum 5,
- no other slice of A has sum greater than (0, 1).

Solution

Programming language used:	C++
Total time used:	2 minutes
Effective time used:	2 minutes
Notes:	*not defined yet*

Task timeline

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

- N is an integer within the range [1..1,000,000];
- each element of array A is an integer within the range [−1,000,000..1,000,000];
- the result will be an integer within the range [−2,147,483,648..2,147,483,647].

Code: 13:51:44 UTC, cpp, final, score: **84** show code in pop-up

```cpp
#include <vector>
#include <algorithm>

int solution(std::vector<int> &A)
{
  if(A.size() == 1)
  {
    return A[0];
  }

  int sumEachIteration = 0;
  int maxSumEachIteration = A[0]; // if 0 will not
  for(int i = 0; i < A.size(); i++)
  {
    for(int j = i; j < A.size(); j++)
    {
      sumEachIteration+=A[j];
      maxSumEachIteration = std::max(maxSumEachIter
    }
    sumEachIteration = 0;
  }

  return maxSumEachIteration;
}
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

Detected time complexity: **O(N) or O(N**3)**

expand all	Example tests	
▶	example	✓ OK
expand all	Correctness tests	
▶	one_element	✓ OK
▶	two_elements	✓ OK
▶	three_elements	✓ OK
▶	simple	✓ OK
▶	extreme_minimum	✓ OK
▶	fifty_random	✓ OK
▶	neg_const	✓ OK
▶	pos_const	✓ OK
expand all	Performance tests	
▶	high_low_1Kgarbage	✓ OK
▶	1Kgarbage_high_low	✓ OK
▶	growing_saw	✓ OK
▶	blocks	✗

TIMEOUT ERROR

Killed. Hard limit reached:

6.000 sec.

▶ growing_negative	✗ **TIMEOUT ERROR** Killed. Hard limit reached: 6.000 sec.