Codility_

CodeCheck Report: trainingXKBJ4V-NQT

Test Name:

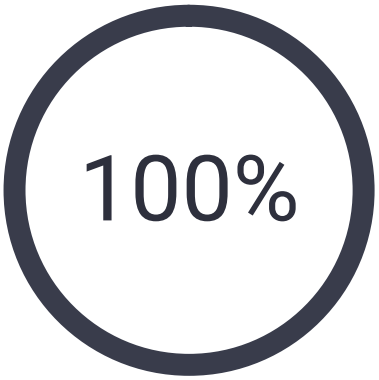
Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxSliceSum Python	1 min	100%

Total score



Tasks Details

1. MaxSliceSum

Easy

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

Task Score	Correctness	Performance
100%	100%	100%

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:

```
def solution(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
```

Solution

Programming language used:	Python
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

Task timeline

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).

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].

12:58:57 12:59:38

Code: 12:59:38 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debugging
2  # print("this is a debug message")
3
4  def solution(A):
5      # Implement your solution here
6      # pass
7      if len(A) == 1:
8          return A[0]
9
10     if len(A) == 0:
11         return 0
12
13     max_ending = 0
14     max_slice = float('-inf')
15
16     for num in A:
17         max_ending = max(num, max_ending
18         max_slice = max(max_slice, max_e
19
20     return max_slice
```

Analysis summary

The solution obtained perfect score.

Analysis

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

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	✔ OK
▶	growing_negative	✔ OK