Codility_

CodeCheck Report: trainingBD6PA4-PS3

Test Name:

Check out Codility training tasks

Summary | Timeline

Tasks summary

Task	Time spent	Score
MaxDoubleSliceSum ⚠ Python	11 min	100%

Total score

Tasks Details

Medium

1. MaxDoubleSliceSum

Find the maximal sum of any double slice.

Task Score	Correctness	Performance
100%	100%	100%

Task description

A non-empty array A consisting of N integers is given.

A triplet (X, Y, Z), such that 0 ≤ X < Y < Z < N, is called a *double slice*.

The *sum* of double slice (X, Y, Z) is the total of A[X + 1] + A[X + 2] + ... + A[Y − 1] + A[Y + 1] + A[Y + 2] + ... + A[Z − 1].

For example, array A such that:

```
A[0] = 3
A[1] = 2
A[2] = 6
A[3] = −1
A[4] = 4
A[5] = 5
A[6] = −1
A[7] = 2
```

contains the following example double slices:

- double slice (0, 3, 6), sum is 2 + 6 + 4 + 5 = 17,
- double slice (0, 3, 7), sum is 2 + 6 + 4 + 5 − 1 = 16,
- double slice (3, 4, 5), sum is 0.

The goal is to find the maximal sum of any double slice.

Write a function:

```
def solution(A)
```

that, given a non-empty array A consisting of N integers, returns the maximal sum of any double slice.

For example, given:

```
A[0] = 3
A[1] = 2
A[2] = 6
A[3] = −1
A[4] = 4
A[5] = 5
A[6] = −1
A[7] = 2
```

the function should return 17, because no double slice of array A has a sum of greater than 17.

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

- N is an integer within the range [3..100,000];
- each element of array A is an integer within the range [−10,000..10,000].

Solution

Programming language used:	Python
Total time used:	11 minutes
Effective time used:	11 minutes
Notes:	*not defined yet*

Task timeline

21:03:46 — 21:14:20

Code: 21:14:19 UTC, py, final, score: **100** — show code in pop-up

```
1  # you can write to stdout for debugging purposes, e.g.
2  # print("this is a debug message")
3
4  def solution(A):
5      # write your code in Python 3.6
6      if not A:
7          return 0
8      N = len(A)
9      if N < 4:
10         return 0
11     if N == 4:
12         return max(A[1], A[2])
13
14     arr_ = [0] * N
15     max_ = 0
16     for i in range(1, N-2):
17         max_ = max(0, A[i]+max_)
18         arr_[i] = max_
19
20     _arr = [0] * N
21     max_ = 0
22     for i in range(N-2, 1, -1):
23         max_ = max(0, A[i]+max_)
24         _arr[i] = max_
25
26     max_double_slice = 0
27     for i in range(0, N-2):
28         max_double_slice = max(max_double_slice, arr_[i] + _ar
29
30     return max_double_slice
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests	
▶	example example test	✔ OK
expand all	**Correctness tests**	
▶	simple1 first simple test	✔ OK
▶	simple2 second simple test	✔ OK
▶	simple3 third simple test	✔ OK
▶	negative all negative numbers	✔ OK
▶	positive all positive numbers	✔ OK
▶	extreme_triplet three elements	✔ OK
expand all	**Performance tests**	
▶	small_random1 random, numbers form -10**4 to 10**4, length = 70	✔ OK
▶	small_random2 random, numbers from -30 to 30, length = 300	✔ OK
▶	medium_range -1000, ..., 1000	✔ OK
▶	large_ones random numbers from -1 to 1, length = ~100,000	✔ OK
▶	large_random random, length = ~100,000	✔ OK
▶	extreme_maximal all maximal values, length = ~100,000	✔ OK
▶	large_sequence many the same small sequences, length = ~100,000	✔ OK