Codility_

CodeCheck Report: trainingYS6WNW-FPR

Test Name:

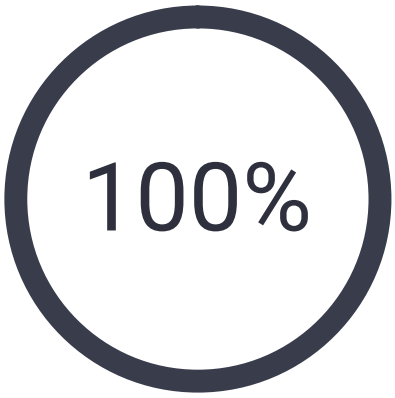
Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
TieRopes Python	5 min	100%

Total score



Tasks Details

1. **TieRopes**

Easy

Tie adjacent ropes to achieve the maximum number of ropes of length >= K.

Task Score	Correctness	Performance
100%	100%	100%

Task description

There are N ropes numbered from 0 to N − 1, whose lengths are given in an array A, lying on the floor in a line. For each I (0 ≤ I < N), the length of rope I on the line is A[I].

We say that two ropes I and I + 1 are *adjacent*. Two adjacent ropes can be tied together with a knot, and the length of the tied rope is the sum of lengths of both

Solution

Programming language used:	Python
Total time used:	5 minutes
Effective time used:	5 minutes

ropes. The resulting new rope can then be tied again.

For a given integer K, the goal is to tie the ropes in such a way that the number of ropes whose length is greater than or equal to K is maximal.

For example, consider K = 4 and array A such that:

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

The ropes are shown in the figure below.

We can tie:

- rope 1 with rope 2 to produce a rope of length A[1] + A[2] = 5;
- rope 4 with rope 5 with rope 6 to produce a rope of length A[4] + A[5] + A[6] = 5.

After that, there will be three ropes whose lengths are greater than or equal to K = 4. It is not possible to produce four such ropes.

Write a function:

```
def solution(K, A)
```

that, given an integer K and a non-empty array A of N integers, returns the maximum number of ropes of length greater than or equal to K that can be created.

For example, given K = 4 and array A such that:

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

the function should return 3, as explained above.

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

- N is an integer within the range [1..100,000];
- K is an integer within the range [1..1,000,000,000];
- each element of array A is an integer within the range [1..1,000,000,000].

Notes: *not defined yet*

Task timeline

14:17:59 14:22:17

Code: 14:22:17 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debuggir
2  # print("this is a debug message")
3
4  def solution(K, A):
5      # Implement your solution here
6      # pass
7      N = len(A)
8      count = 0
9      current_length = 0
10
11     for i in range(N):
12         current_length += A[i]
13         if current_length >= K:
14             count += 1
15             current_length = 0
16
17     return count
18
```

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**	
▶	single single element	✔ OK
▶	double two elements	✔ OK
▶	small_functional small functional tests	✔ OK
▶	small_random small random sequences length	✔ OK

= ~100

expand all

Performance tests

▶	medium_random chaotic medium sequences length = ~5,000	✔ OK
▶	large_range large range test, length = ~100,000	✔ OK
▶	large_answer test with large answer, length = ~100,000	✔ OK
▶	small_answer test with large answer, length = ~100,000	✔ OK