Codility_

CodeCheck Report: trainingTQSNKN-364

Test Name:

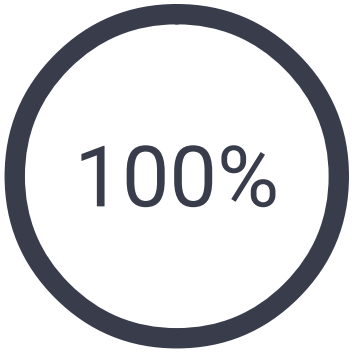
Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
ScooterRoad C++	18 min	100%

Total score



Tasks Details

Medium

1. ScooterRoad

Calculate the minimum time that you need to get through the diversified road to your work.

Task Score	Correctness	Performance
100%	100%	100%

Task description

You have to be at your work as soon as possible. The road on your route to work may consist of two types of surface: asphalt or sand. To simplify the description, it will be denoted by a string R consisting only of the letters: "A" for an asphalt segment and "S" for a sand segment. All segments represent the same distance. For example, R = "SAAS" describes a road comprising of sand, asphalt, asphalt and sand segments.

When you go on foot, you need 20 minutes to pass through an asphalt segment and 30 minutes through a sand segment. You also have an electric scooter, which needs 5 minutes to pass through an asphalt segment and 40 minutes through a sand segment.

You start your journey on the scooter, but at any point you can get off the scooter and go on foot for the rest of the journey. What is the shortest time in which you can get to work?

Write a function:

```
int solution(string &R);
```

that, given a string R of length N, representing the road to work, returns the minimum time that you need to get to work.

Solution

Programming language used:	C++
Total time used:	18 minutes
Effective time used:	18 minutes
Notes:	*not defined yet*

Task timeline

11:13:59 11:31:08

Code: 11:31:08 UTC, cpp, final, score: **100** show code in pop-up

```
1  // you can use includes, for example:
2  #include <bits/stdc++.h>
```

Examples:

1. Given R = "ASAASS", your function should return 115. You ride on the scooter over the first four segments ("ASAA") in 5 + 40 + 5 + 5 = 55 and then you go on foot through "SS" in 30 + 30 = 60. Altogether, your journey will take 55 + 60 = 115.

2. Given R = "SSA", the function should return 80. You do not ride on the scooter at all, and you go on foot in 30 + 30 + 20 = 80.

3. Given R = "SSSSAAA", the function should return 175. You ride on the scooter all the time in 40 + 40 + 40 + 40 + 5 + 5 + 5 = 175.

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

- N is an integer within the range [1..100,000];
- string R is made only of the characters 'S' and/or 'A'.


```
3
4   // you can write to stdout for debugging purposes,
5   // cout << "this is a debug message" << endl;
6
7   int solution(string &R) {
8       // Implement your solution here
9       vector<int>sco;
10      vector<int>foot;
11      sco.push_back(0);
12      foot.push_back(0);
13      for(int i=0;i<R.size();i++)
14      {
15          if(R[i]=='A')
16          {
17              sco.push_back(sco.back()+5);
18              foot.push_back(foot.back()+20);
19          }
20          else{
21              sco.push_back(sco.back()+40);
22              foot.push_back(foot.back()+30);
23          }
24      }
25      int ans=min(sco.back(),foot.back());
26      // int s=0;
27      int f=foot.back();
28
29      for(int i=0;i<R.size();i++)
30      {
31          ans=min(ans,sco[i+1]+(f-foot[i+1]));
32      }
33      return ans;
34  }
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests	
▶	example1 First example test.	✓ OK
▶	example2 Second example test.	✓ OK
▶	example3 Third example test.	✓ OK
expand all	Correctness tests	
▶	very_short_road N = 1.	✓ OK
▶	short_road N <= 3.	✓ OK
▶	only_scooter Only scooter is used.	✓ OK
▶	only_walking Scooter is not used at all.	✓ OK
▶	all_asphalt_first Road can be described by "AA...ASS...S". N <= 200.	✓ OK
▶	small_random_change_point Getting off the scooter at a random	✓ OK

position. N <= 200.

expand all

Performance tests

▶	medium_random Medium random tests. N <= 10,000.	✓ OK
▶	medium_random_change_point Medium tests, getting off the scooter at a random position. N <= 10,000.	✓ OK
▶	big_random Big random tests.	✓ OK
▶	big_random_change_point Big tests, getting off the scooter at a random position.	✓ OK
▶	big_corner_cases Big tests with corner cases.	✓ OK