Codility_

Candidate Report: trainingQRA2VZ-S7J

Check out Codility training tasks

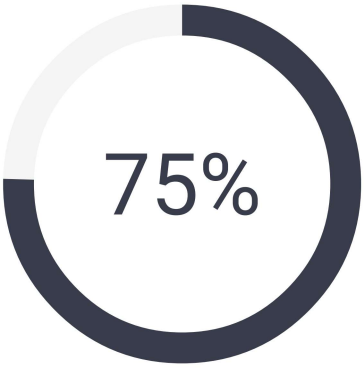
Test Name:

Summary Timeline Feedback

Tasks summary

Task	Time spent	Score
GenomicRangeQuery C#	1 min	75%

Total score



Tasks Details

Medium

1. GenomicRangeQuery

Find the minimal nucleotide from a range of sequence DNA.

Task Score	Correctness	Performance
75%	100%	33%

Task description

A DNA sequence can be represented as a string consisting of the letters A, C, G and T, which correspond to the types of successive nucleotides in the sequence. Each nucleotide has an *impact factor*, which is an integer. Nucleotides of types A, C, G and T have impact factors of 1, 2, 3 and 4, respectively. You are going to answer several queries of the form: What is the minimal impact factor of nucleotides contained in a particular part of the given DNA sequence?

The DNA sequence is given as a non-empty string S = `S[0]S[1]...S[N-1]` consisting of N characters. There are M queries, which are given in non-empty arrays P and Q, each consisting of M integers. The K-th query ($0 \le K < M$) requires you to find the minimal impact factor of nucleotides contained in the DNA sequence between positions P[K] and Q[K] (inclusive).

For example, consider string S = `CAGCCTA` and arrays P, Q such that:

```
P[0] = 2    Q[0] = 4
P[1] = 5    Q[1] = 5
P[2] = 0    Q[2] = 6
```

The answers to these M = 3 queries are as follows:

Solution

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

Task timeline

02:52:02 02:52:21

Code: 02:52:20 UTC, cs, final, score: 75

show code in pop-up

- The part of the DNA between positions 2 and 4 contains nucleotides G and C (twice), whose impact factors are 3 and 2 respectively, so the answer is 2.
- The part between positions 5 and 5 contains a single nucleotide T, whose impact factor is 4, so the answer is 4.
- The part between positions 0 and 6 (the whole string) contains all nucleotides, in particular nucleotide A whose impact factor is 1, so the answer is 1.

Write a function:

```
class Solution { public int[] solution(string S, int[] P, int[] Q); }
```

that, given a non-empty string S consisting of N characters and two non-empty arrays P and Q consisting of M integers, returns an array consisting of M integers specifying the consecutive answers to all queries.

Result array should be returned as an array of integers.

For example, given the string S = CAGCCTA and arrays P, Q such that:

```
P[0] = 2    Q[0] = 4
P[1] = 5    Q[1] = 5
P[2] = 0    Q[2] = 6
```

the function should return the values [2, 4, 1], as explained above.

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

- N is an integer within the range [1..100,000];
- M is an integer within the range [1..50,000];
- each element of arrays P, Q is an integer within the range [0..N − 1];
- P[K] ≤ Q[K], where 0 ≤ K < M;
- string S consists only of upper-case English letters A, C, G, T.


```
1  using System;
2  // you can also use other imports, for example:
3  // using System.Collections.Generic;
4
5  // you can write to stdout for debugging purposes, e.g.
6  // Console.WriteLine("this is a debug message");
7
8  class Solution {
9        public int[] solution(String S, int[] P, int[] Q)
10         {
11             int M = P.Length;
12             int[] retVal = new int[M];
13
14             for (int i = 0; i < M; i++)
15             {
16                 var cnt = Q[i] - P[i] + 1;
17                 int a = S.IndexOf('A', P[i], cnt);
18                 if (S.IndexOf('A', P[i], cnt) > -1)
19                 {
20                     retVal[i] = 1;
21                 }
22                 else if (S.IndexOf('C', P[i], cnt) > -1)
23                 {
24                     retVal[i] = 2;
25                 }
26                 else if (S.IndexOf('G', P[i], cnt) > -1)
27                 {
28                     retVal[i] = 3;
29                 }
30                 else if (S.IndexOf('T', P[i], cnt) > -1)
31                 {
32                     retVal[i] = 4;
33                 }
34
35
36             }
37             return retVal;
38         }
39 }
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

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

expand all	Example tests	
▶	example example test	✓ OK
expand all	Correctness tests	
▶	extreme_sinlge single character string	✓ OK
▶	extreme_double double character string	✓ OK
▶	simple simple tests	✓ OK
▶	small_length_string small length simple string	✓ OK
▶		

small_random small random string, length = ~300	✓ OK	
expand all	Performance tests	
▶ almost_all_same_letters GGGGGG..??..GGGGGG..??..GGGGGG	✗ TIMEOUT ERROR	running time: 0.596 sec., time limit: 0.208 sec.
▶ large_random large random string, length	✓ OK	
▶ extreme_large all max ranges	✗ TIMEOUT ERROR	running time: 4.684 sec., time limit: 0.208 sec.