codility

Check out Codility training tasks

Candidate Report: Anonymous

Test Name:

Summary Timeline

Test Score

100 out of 100 points

100%

Tasks in Test

	Time Spent	Task Score
GenomicRangeQuery Submitted in: Python	14 min	100%

TASKS DETAILS

MEDIUM

1. GenomicRangeQuery Find the minimal nucleotide from a range of sequence DNA.	Task Score	Correctness	Performance
	100%	100%	100%

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 ≤ 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:

- 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:

Solution

Programming language used:	Python
Total time used:	14 minutes
Effective time used:	14 minutes
Notes:	*not defined yet*

Task timeline

06:21:40 06:34:57

Code: 06:34:57 UTC, py, final, score: **100**

show code in pop-up

```
"""
Solution for task 05.2 GenomicRangeQuery
"""
def solution(s, p, q):
    a_prefix = [0 for _ in range(len(s) + 1)]
    c_prefix = a_prefix.copy()
    g_prefix = a_prefix.copy()
    for i, nucleotide in enumerate(s):
        a_prefix[i + 1] = a_prefix[i] + (1 if nucleotide =
def solution(S, P, 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`.

```
        c_prefix[i + 1] = c_prefix[i] + (1 if nucleotide =
        g_prefix[i + 1] = g_prefix[i] + (1 if nucleotide =
    result = []
    for left, right in zip(p, q):
        if a_prefix[right + 1] - a_prefix[left] > 0:
            result.append(1)
        elif c_prefix[right + 1] - c_prefix[left] > 0:
            result.append(2)
        elif g_prefix[right + 1] - g_prefix[left] > 0:
            result.append(3)
        else:
            result.append(4)
    return result
```

Analysis summary

The solution obtained perfect score.

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	✓ OK
▶	large_random large random string, length	✓ OK
▶	extreme_large all max ranges	✓ OK