Codility_

Candidate Report: trainingDJXE4D-XQN

Check out Codility training tasks

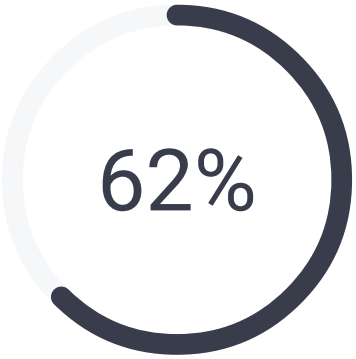
Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
GenomicRangeQuery Python ⚠	20 min	62%

Total score



Tasks Details

1. **GenomicRangeQuery**
Medium
Find the minimal nucleotide from a range of sequence DNA.

Task Score	Correctness	Performance
62%	100%	0%

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:

Solution

Programming language used:	Python
Total time used:	20 minutes
Effective time used:	20 minutes
Notes:	*not defined yet*

Task timeline

```
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:

```
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.

Code: 10:33:52 UTC, py, final, score: **62** show code in pop-up

```python
# you can write to stdout for debugging purposes, e.g.
# print("this is a debug message")

def solution(S, P, Q):
    # write your code in Python 3.6
    tmp = []
    M = len(P)
    i = 0
    while i<M:
        st = P[i]
        ed = Q[i]
        sub = S[st:ed+1]
        dic = {}
        for ch in sub:
            dic[ch]=1
            if len(dic)==4:
                break
        #print(sub, dic)
        num = 4
        if 'A' in dic:
            num = 1
        elif 'C' in dic:
            num = 2
        elif 'G' in dic:
            num = 3
        tmp.append(num)
        i = i + 1

    return tmp
```

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 Killed. Hard limit

		reached: 6.000 sec.
▶	large_random large random string, length	✗ **TIMEOUT ERROR** Killed. Hard limit reached: 6.000 sec.
▶	extreme_large all max ranges	✗ **TIMEOUT ERROR** Killed. Hard limit reached: 7.000 sec.