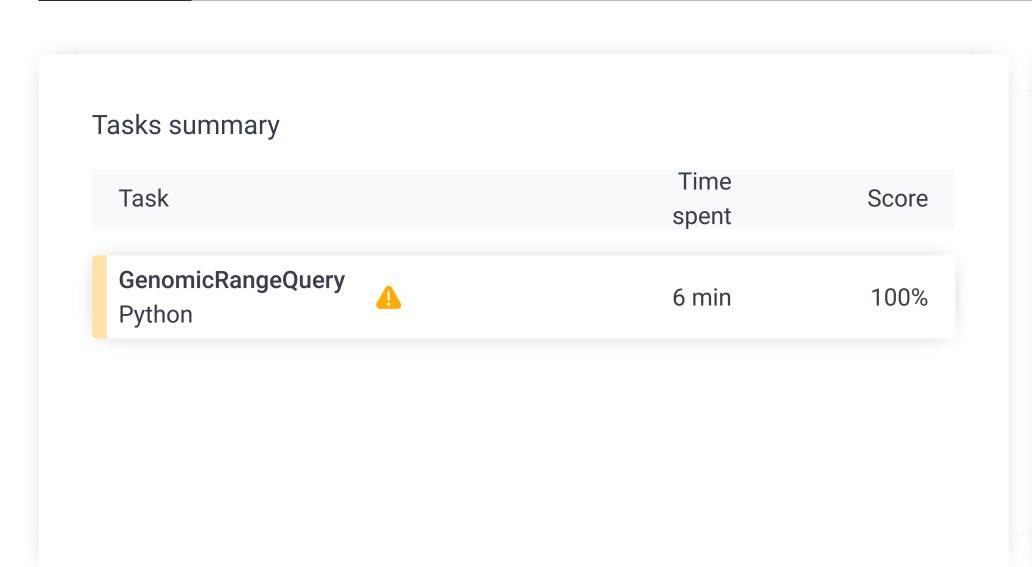
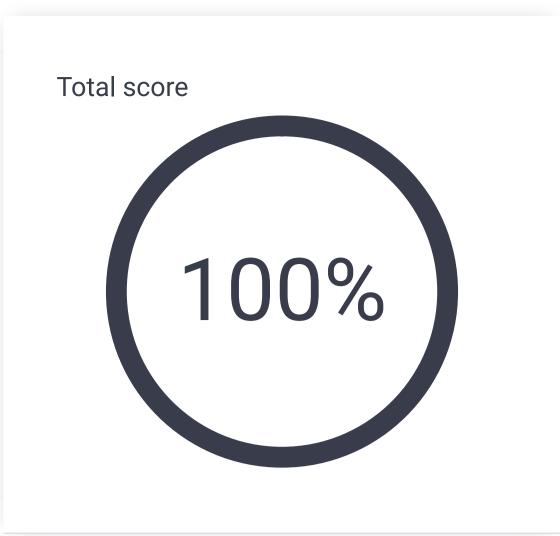
100%

### CodeCheck Report: trainingFS9J46-2AN

Test Name:

Summary Timeline





#### **Tasks Details**



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

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

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

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

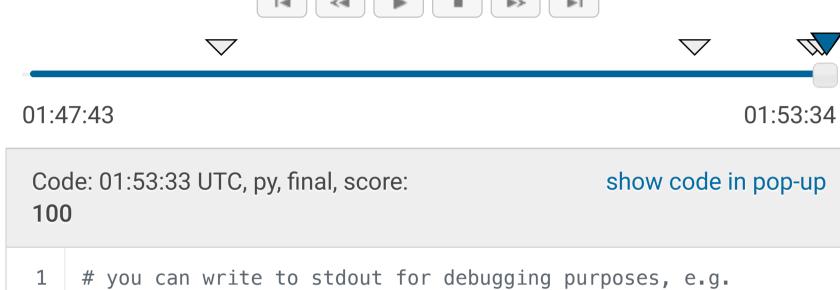
- N is an integer within the range [1..100,000];
- M is an integer within the range [1..50,000];
- each element of arrays P and Q is an integer within the range [0..N - 1];
- $P[K] \le Q[K]$ , where  $0 \le K < M$ ; • string S consists only of upper-case English letters A, C, G, T.

Copyright 2009-2022 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

#### Solution

Programming language used: Python Total time used: ? 6 minutes Effective time used: 6 minutes ? not defined yet Notes:

Task timeline



```
# print("this is a debug message")
     def solution(S, P, Q):
         # write your code in Python 3.6
         M = len(P)
         if M != len(Q):
             return -1
         N = len(S)
 9
         dict_ = {
10
11
             'A' : 1,
12
             'C' : 2,
             'G' : 3,
13
             'T' : 4
14
15
16
17
         arr=[]
18
         for el in S:
19
             arr.append(dict_[el])
20
         global_minimum = min(arr)
21
22
         if N == 1:
             return ([arr[0]] * M)
23
24
         else:
             dict1 = \{\}
25
26
             i = 0
27
             for el in P:
28
                 if el not in dict1.keys():
29
                     dict1[el] = []
30
                 dict1[el].append(i)
31
                 i += 1
             dict2 = \{\}
32
33
             i = 0
34
             for el in Q:
35
                 if el not in dict2.keys():
36
                     dict2[el] = []
37
                 dict2[el].append(i)
38
                 i += 1
39
40
             min_ = max(arr) + 1
41
             min_before_reset = min_
42
             dict_ = {}
             queries = {}
43
44
             for i in range(N):
45
                 min_ = min(min_, arr[i])
46
47
                 if i in dict1.keys() and i in dict2.keys():
48
                     list_ = []
49
                     for el1 in dict1[i]:
50
                         for el2 in dict2[i]:
51
                             if el1 == el2:
52
                                  queries[el1] = arr[i]
53
                                  list_.append(el1)
54
                     list1 = []
                     for el1 in dict1[i]:
55
56
                         if el1 not in list_:
57
                             list1.append(el1)
58
                     dict1[i] = list1.copy()
59
                     list2 = []
                     for el2 in dict2[i]:
60
61
                         if el2 not in list_:
62
                             list2.append(el2)
63
                     dict2[i] = list2.copy()
64
65
                 if i in dict2.keys():
66
                     list_ = []
67
                     for el in dict2[i]:
68
                         if el in dict_.keys():
                             dict_[el] = min(min_, dict_[el])
69
                             queries[el] = dict_[el]
70
71
                             list_.append(el)
72
                     for el in list_:
73
                         dict_.pop(el)
74
75
                 if i in dict1.keys():
                     if min_before_reset != min_:
76
77
                         for key in dict_.keys():
78
                             dict_[key] = min(min_, dict_[key])
79
                     for el in dict1[i]:
                         if arr[i] == global_minimum:
80
81
                             queries[el] = global_minimum
82
                         else:
83
                             dict_[el] = arr[i]
84
                             min_before_reset = min_
85
                             min_ = arr[i]
86
87
                 if min_ == global_minimum:
                     for key in dict_.keys():
88
                         queries[key] = global_minimum
89
                     dict_ = ({}).copy()
90
91
92
             result = [ queries[i] for i in range(M)]
93
94
             return result
```

# Analysis summary

The solution obtained perfect score.

GGGGGG..??..GGGGGG

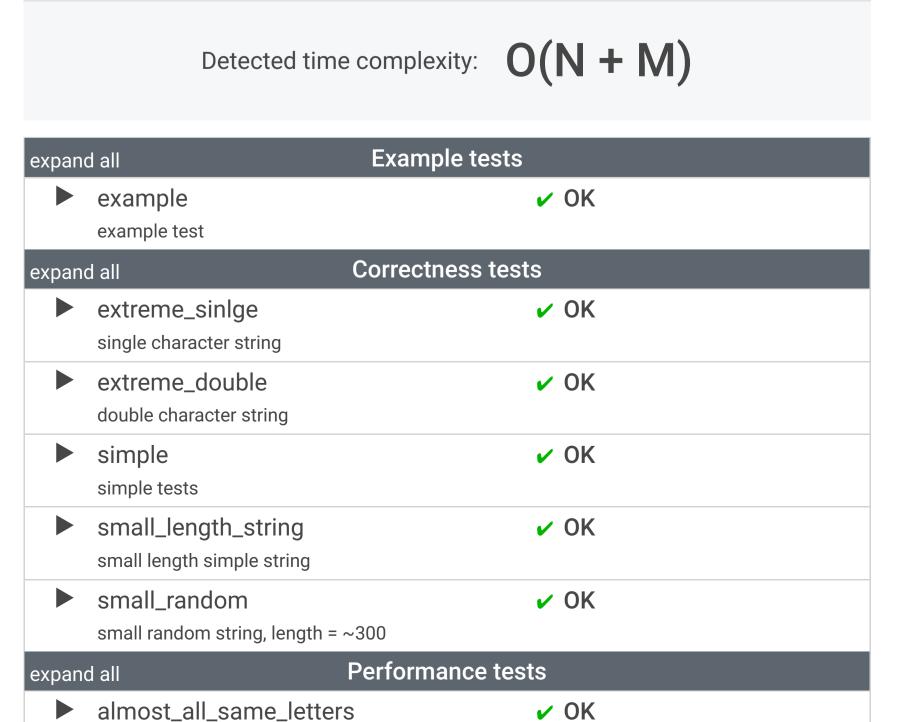
large\_random

extreme\_large

all max ranges

large random string, length

# Analysis



✓ OK

✓ OK

✓ OK