

closed

Demo ticket

Session**ID:** demoTN49M8-AYD**Time limit:** 120 min.**Status:** closed**Started on:** 2014-01-06 16:54 UTC

Score:

100

of 100



☆☆ 1. GenomicRangeQuery

Find the minimal nucleotide from a range of sequence DNA.

score: 100 of 100

**Task description**

A non-empty zero-indexed string S is given. String S consists of N characters from the set of upper-case English letters A, C, G, T . This string actually represents a DNA sequence, and the upper-case letters represent single nucleotides. You are also given non-empty zero-indexed arrays P and Q consisting of M integers. These arrays represent queries about minimal nucleotides. We represent the letters of string S as integers $1, 2, 3, 4$ in arrays P and Q , where $A = 1, C = 2, G = 3, T = 4$, and we assume that $A < C < G < T$. Query K requires you to find the minimal nucleotide from the range $(P[K], Q[K])$, $0 \leq P[i] \leq Q[i] < N$. For example, consider string $S = \text{GACACCATA}$ and arrays P, Q such that:

```
P[0] = 0    Q[0] = 8
P[1] = 0    Q[1] = 2
P[2] = 4    Q[2] = 5
P[3] = 7    Q[3] = 7
```

The minimal nucleotides from these ranges are as follows:

- (0, 8) is **A** identified by 1,
- (0, 2) is **A** identified by 1,
- (4, 5) is **C** identified by 2,
- (7, 7) is **T** identified by 4.

Write a function:

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

that, given a non-empty zero-indexed string S consisting of N characters and two non-empty zero-indexed arrays P and Q consisting of M integers, returns an array consisting of M characters specifying the consecutive answers to all queries. The sequence should be returned as:

- a Results structure (in C), or
- a vector of integers (in C++), or
- a Results record (in Pascal), or
- an array of integers (in any other programming language).

For example, given the string $S = \text{GACACCATA}$ and arrays P, Q such that:

```
P[0] = 0    Q[0] = 8
P[1] = 0    Q[1] = 2
P[2] = 4    Q[2] = 5
P[3] = 7    Q[3] = 7
```

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

Assume that:

- N is an integer within the range $[1..100,000]$;
- M is an integer within the range $[1..50,000]$;
- each element of array P, Q is an integer within the range $[0..N - 1]$;
- $P[i] \leq Q[i]$;
- string S consists only of upper-case English letters A, C, G, T .

Complexity:

- expected worst-case time complexity is $O(N+M)$;

Solution**Programming language used:** C#**Total time used:** 1 minutes

(?)

Effective time used: 1 minutes

(?)

Notes: correct functionality and scalability**Task timeline**

What is it? (?)



16:54:15

16:54:43

Code: 16:54:43 UTC, cs, final, score: 100.00

```
01. using System;
02. // you can also use other imports, for
   // example:
03. // using System.Collections.Generic;
04. class Solution {
05.     public int[] solution(string S,
        int[] P, int[] Q)
06.     {
07.         // write your code in C# with
        // .NET 2.0
08.         if (S.Length < 1 || S.Length >
            100000
09.             || P.Length < 1 || P.Length
            > 50000
10.             || Q.Length < 1 || Q.Length
            > 50000
11.             || P.Length != Q.Length)
            throw new
                ArgumentOutOfRangeException();
12.
13.
14.         var nucleotideCount = new
            int[S.Length + 1, 4];
15.         for (var count = 0; count <
            S.Length; count++)
16.         {
17.             // Prefix Sums should start
            // at 0 index = 0 and length
            // + 1 with the total values
18.             if (count > 0)
19.             {
20.                 // Skip adding the first
                // row at index 0 which
                // contains only zeros
21.                 for (var index = 0;
                    index < 4; index++)
22.                 {
23.                     nucleotideCount[count
```

- expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.
Copyright 2009–2013 by Codility Limited. All Rights Reserved.
Unauthorized copying, publication or disclosure prohibited.

```
24.
25.
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75. }
```

```

    + 1, index] +=
    nucleotideCount[count,
    index];
    }
    }
    switch (S[count])
    {
        case 'A':
            nucleotideCount[count
            + 1, 0]++;
            break;
        case 'C':
            nucleotideCount[count
            + 1, 1]++;
            break;
        case 'G':
            nucleotideCount[count
            + 1, 2]++;
            break;
        case 'T':
            nucleotideCount[count
            + 1, 3]++;
            break;
    }
    }

    var result = new int[P.Length];
    for (var count = 0; count <
    P.Length; count++) {
        if(P[count] == Q[count])
        {
            switch(S[P[count]]) {
                case 'A':
                    result[count] =
                    1;
                    break;
                case 'C':
                    result[count] =
                    2;
                    break;
                case 'G':
                    result[count] =
                    3;
                    break;
                case 'T':
                    result[count] =
                    4;
                    break;
            }
        } else {
            for(var index = 0; index
            < 4; index++) {
                if((nucleotideCount[Q[count
                + 1, index] -
                nucleotideCount[P[count]
                index]) > 0) {
                    result[count] =
                    index + 1;
                    break;
                }
            }
        }
    }
    }
    return result;
}
```

Analysis



Detected time complexity:

O(N + M)

[Get account](#)

test	time	result
example	0.080 s.	OK
example test		
extreme_sinlge	0.080 s.	OK

single character string		
extreme_double double character string	0.080 s.	OK
simple simple tests	0.080 s.	OK
small_length_string small length simple string	0.080 s.	OK
small_random small random string, length = ~300	0.080 s.	OK
almost_all_same_letters GGGGGG..??..GGGGGG..??..GGGGGG	0.090 s.	OK
large_random large random string, length	0.120 s.	OK
extreme_large all max ranges	0.120 s.	OK