Training ticket

Session

ID: trainingHXJSQZ-6Y7
Time limit: 120 min.

Status: closed

Created on: 2016-06-11 10:38 UTC
Started on: 2016-06-11 10:38 UTC
Finished on: 2016-06-11 10:38 UTC

Tasks in test	Correctness	Performance	Task score
1 GenomicRangeQuery Submitted in: Java	100%	100%	100%

Test score

100%

100 out of 100 points

MEDIUM

1. GenomicRangeQuery

Find the minimal nucleotide from a range of sequence DNA.

score: 100 of 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

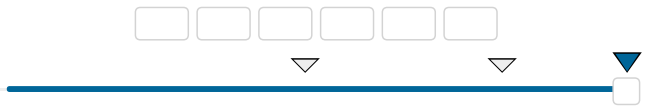
Programming language used: Java

Total time used: 1 minutes

Effective time used: 1 minutes

Notes: *not defined yet*

Task timeline



10:38:06 10:38:31

Code: 10:38:31 UTC, java, final, score: **100**

show code in pop-up

```
1  // you can also use imports, for example:
2  // import java.util.*;
3
4  // you can write to stdout for debugging purposes, e.g.
5  // System.out.println("this is a debug message");
6
7  class Solution {
8      public int[] solution(String S, int[] P, int[] Q) {
9          char[] gen = S.toCharArray();
10         int[] geni = new int[gen.length];
11
12         for (int i = 0; i < gen.length; i++) {
13             switch (gen[i]) {
14                 case 'A': geni[i] = 1;
15                     break;
```

```
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 integers 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 = `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.

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 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`.

Complexity:

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


```
16                  case 'C': geni[i] = 2;
17                      break;
18                  case 'G': geni[i] = 3;
19                      break;
20                  case 'T': geni[i] = 4;
21                      break;
22              }
23          }
24          int[] res = new int[P.length];
25          int[][] map = new int[4][geni.length];
26          int[] mapi = new int[4]; // max position of the
27
28          int idx = 0;
29          for (int i = 0; i < geni.length; i++) {
30              idx = geni[i]-1;
31              map[idx][mapi[idx]++] = i;
32          }
33
34          for (int i = 0; i < P.length; i++) {
35              for (int j = P[i]; j <= Q[i]; j++) {
36
37                  for (int k = 0; k < 4; k++) {
38                      for (int l = 0; l < mapi[k]; l++) {
39                          if ( P[i] <= map[k][l] && map[l
40                              res[i] = (k+1);
41                              break;
42                          }
43                      }
44                      if (res[i] > 0) {
45                          break;
46                      }
47                  }
48                  if (res[i] > 0) {
49                      break;
50                  }
51              }
52          }
53          return res;
54      }
55  }
```

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