CS1020E | Lab 11 | Exercise 1

DNA Subsequences

Objectives

The focus of this exercise is algorithm efficiency and time complexity.

Problem Description

Given a DNA sequence and a number of substrings, write a program to find the number of occurrences of each substring in the DNA sequence.

Add your code only to the parts of the file indicated. Do not modify any other part of the given code, and do not add new files.

Inputs

The first line of the input contains *N* (1 <= *N* <= 500) and *K* (1 <= *K* <= 6), where *N* is the length of that DNA and *K* is the length of the substring that we are interested in. The second line contains the DNA sequence of length *N*. The DNA sequence is a string consisting of only characters 'A', 'C', 'G' and/or 'T'. The next line contains an integer *Q* (1 <= *Q* <= 1,000), which denotes the number of substrings that we are interested in (*Q* is also called the number of queries). The next *Q* lines contain the substrings, each of length *K*.

Outputs

There are *Q* lines in the output. Each line contains the number of occurrences of the corresponding substring in the DNA sequence.

Sample Input 1

```
6 2
ACGTAC
6
AC
CG
AT
GT
TA
CA
```

Sample Output 1

```
2
1
0
1
```

```
1
0
```

Explanation 1

ACGT**AC** There are 2 **AC** in our DNA sequence.
A**CG**TAC There is 1 **CG** in our DNA sequence.
ACGTAC There is 0 **AT** in our DNA sequence.
AC**GT**AC There is 1 **GT** in our DNA sequence.
ACG**TA**C There is 1 **TA** in our DNA sequence.
ACGTAC There is 0 **CA** in our DNA sequence.

Sample Input 2

```
5 3
ACACA
1
ACA
```

Sample Output 2

```
2
```

Explanation 2

ACACA AC**ACA** There are 2 **ACA (overlapping)** in our DNA sequence.

Additional Requirements

An efficient program is required. Specifically, you will get **at most 60% of the marks** if the total time complexity of your solution for *all* the queries is worse than O(*NK* + *QK*). You can get **100% of the marks** only if your program's total time complexity for all the queries is equal to or better than O(*NK* + *QK*). To achieve that, each query must take at most O(*K*) time.

(Hint: Think of a data structure that allows the search/query of a key in O(1) time. Then think about how to map each substring of the DNA sequence to some location in the data structure *without collision*.)

Submission

You need to submit your completed **`DNA.cpp`** to CodeCrunch (https://codecrunch.comp.nus.edu.sg/) before the specified deadline. We will take only your latest submission.

Late submissions will not be accepted. The submission system in CodeCrunch will automatically close at the deadline.