# **Problem Solving**

**Lab #12 (Chapter 11)** 

**Department of Computer Engineering Kyung Hee University** 

### Distinct Subsequences

### PC/UVa IDs: 111102/10069, Popularity: B, Success rate: average Level: 3

- A subsequence of a given sequence S consists of S with zero or more elements deleted. Formally, a sequence  $Z = z_1 z_2 \dots z_k$  is a subsequence of  $X = x_1 x_2 \dots x_m$  if there exists a strictly increasing sequence  $< i_1, i_2, \dots, i_k >$  of indices of X such that for all  $j = 1, 2, \dots, k$ , we have  $x_{ij} = z_j$ . For example, Z = bcdb is a subsequence of X = abcbdab with corresponding index sequence < 2, 3, 5, 7 >.
- Your job is to write a program that counts the number of occurrences of Z in X as a subsequence such that each has a distinct index sequence.

### Input

• The first line of the input contains an integer N indicating the number of test cases to follow. The first line of each test case contains a string X, composed entirely of lowercase alphabetic characters and having length no greater than 10,000. The second line contains another string Z having length no greater than 100 and also composed of only lowercase alphabetic characters. Be assured that neither Z nor any prefix or suffix of Z will have more than 10<sup>100</sup> distinct occurrences in X as a subsequence.

### Output

For each test case, output the number of distinct occurrences of Z in X as a subsequence.

Output for each input set must be on a separate line.

Sample Input
2
5
babgbag
bag
rabbbit
rabbit

## What you have to do

### 1. Analysis of the problem

Understand what the problem is.

### 2. Make test cases

- Make test cases more than 5 for the problem
- A test case is usually a single step, or occasionally a sequence of steps, to test the correct behavior and functionalities.
- Test cases can be used to find some error in your partner program

### 3. Design

Design a solution strategy for solving the problem and write its pseudo code

### 4. Implement the program

Implement the program according to the pseudo code

### 5. Test

- Test implemented program by using the test cases
- \* Submit the pseudo code and the implemented program to the assignment submission board

# SOLUTION

# UNDERSTANDING PROBLEM

# Examples: Distinct Subsequences

- Exampe 1: X = babgbag, Z = bag
  - babgbag
  - **ba**bgba**g**
  - **b**abgb**ag**
  - babgbag
  - babgbag
  - So, there are 5 distinct subsequences.
- Example 2: X = rabbbit, z = rabbit
  - rabbbit
  - <u>ra</u>b<u>b</u>b<u>it</u>
  - <u>ra</u>bb<u>bit</u>
  - So, there are 3 distinct subsequences

# DESIGN

# Design Sketch

- This is an simple instance of dynamic programming problems.
  - What is the recurrence relation in this problem?
- This is also an instance of counting problems and the maximum number of occurrences is 10<sup>100</sup>.
  - We need to use a Big Integer representation.
  - For today's lab, design and implement class BigInt for representing big integers.

# Dynamic Programming

- Let X be the sequence, and Y be the subsequence you're looking for.
- Let a[i][j] be the number of times the sequence (Y1 ... Yi) appears inside the string (X1 ... Xj).
- Initially, a[0][j] = 1 for all j, as the null string appears once in any string.
   Also a[i][0] = 0 for all i > 0 as a non-empty substring of the subsequence cannot appear in an empty string.
- Iterate through a[][] row-by-row with the following recurrence:
  - If Yi == Xj, a[i][j] = a[i-1][j-1] + a[i][j-1]
     (You have 1 new match for every occurrence of Y1...Yi in X1...Xj)
  - If Yi != Xj, a[i][j] = a[i][j-1]
     (No new occurrence, so copy previous value)
- At the end, output the last cell of the array

### Pseudo Code

Initialization

```
for j = 0 to X.length: a[0][j] = 1;
for i = 1 to Z.length: a[i][0] = 0;
```

Recurrence Steps

```
for i = 1 to Z.length

for j = 1 to X.length

if X[j] == Z[i]:

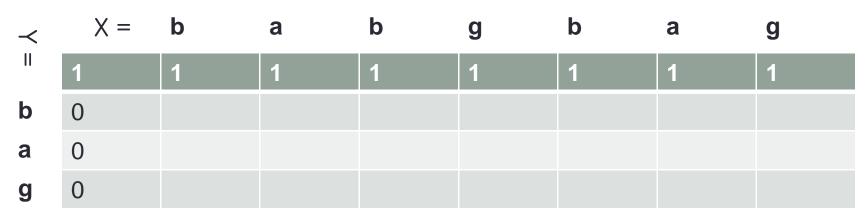
a[i][j] = a[i][j-1] + a[i-1][j-1];

else

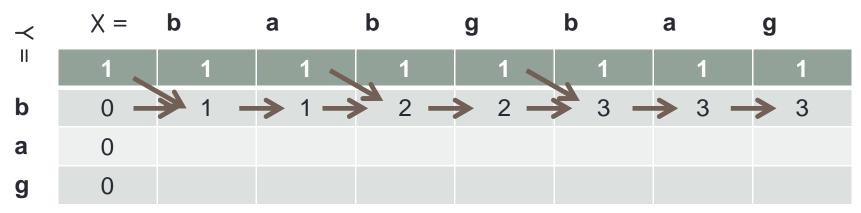
a[i][j] = a[i][j-1];
```

Goal: we get the result at index a[Z.length][X.length];

# Example



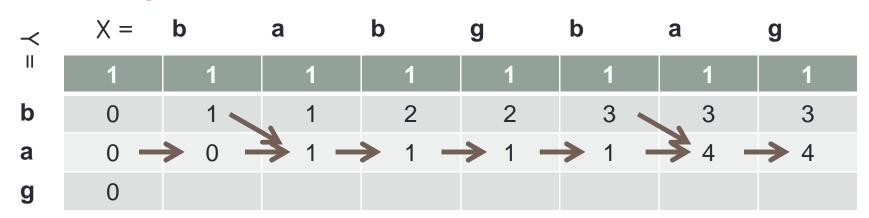
Initial Step of matrix a



After 1st element scan

if X (j) == Y (i): 
$$a[i][j] = a[i][j-1] + a[i-1][j-1];$$
 otherwise:  $a[i][j] = a[i][j-1];$ 

# Example cont...



After 2<sup>nd</sup> element scan

~	X =	b	а	b	g	b	a	g
Ш	1	1	1	1	1	1	1	1
b	0	1	1	2	2	3	3	3
a	0	0	1	1	1	1	4	4
g	0 —	<b>→</b> 0 -	1 > 0 -	<b>&gt;</b> 0 -	<b>9</b> 1 -	<b>→</b> 1 -	<b>→</b> 1 <del>〈</del>	5

After 3<sup>rd</sup> element scan

if X (j) == Y (i): 
$$a[i][j] = a[i][j-1] + a[i-1][j-1];$$
 otherwise:  $a[i][j] = a[i][j-1];$  Finally we find output 5

# C Strings Indexing Problem

- Indices of C strings range from 0 to string length 1.
- So, we need to be careful for using indices when comparing strings. We have to refer to X[j-1] for jth character of string X if X is represented as C string.

```
for i = 1 to Z.length

for j = 1 to X.length

if X[j-1] == Z[i-1]:

a[i][j] = a[i] [j-1] + a[i-1] [j-1];

else

a[i] [j] = a[i][j-1];
```

# Another Solution: C Strings Indexing Problem

- Another approach to solve this problem is to use different interpretation for matrix a.
  - Old: Let a[i][j] be the number of times the sequence (Y1 ... Yi) appears inside the string (X1 ... Xj).
    - Assumption: The strings are stored from index 1.
  - New: Let a[i][j] be the number of times the sequence (Y0 ... Yi) appears inside the string (X0 ... Xj).
    - Assumption: The strings are stored from index 0.

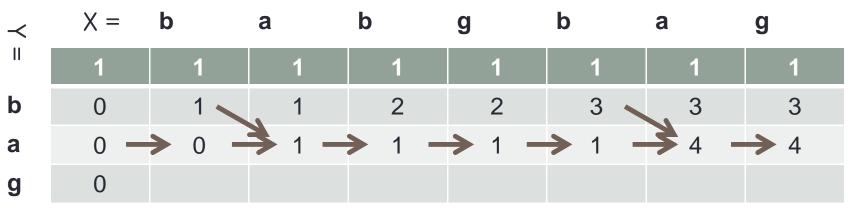
- The new definition requires to change the initialization.
- According to the new definition, a[0][j] should be interpreted as # of times the sequence (Y0) appears in inside the string (X0 ... Xj). So, we have

```
a[0][0] = (X[0] == Z[0]) ? 1:0;
for (int j = 1; j < X.length; j++)
a[0][j] = a[0][j-1] + ((X[j] == Z[0]) ? 1:0);
```

Similarly, a[i][0] represents # of times the sequence (Y0 .. Yi) appears in inside the string (X0). It is 0 when i > 0. So, we have for (int i = 1; i < Z.length; i++) a[i][0] = 0;</li>

# Optimization: Required Memory for Dynamic Programming

- This solution requires a dynamic programming matrix whose size is Z.length \* X.length. Can we reduce its size?
- It is an interesting observation that a[i][j] only refers to the previous row, a[i-1][]. At any time, we are using only two rows: the previous and the current.



After 2<sup>nd</sup> element scan

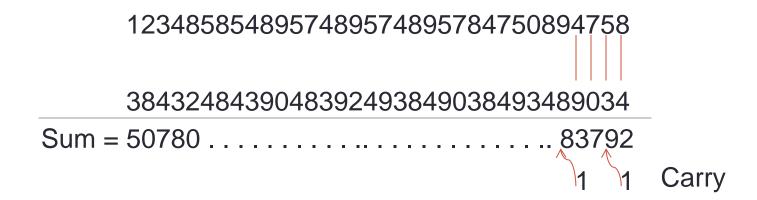
- So we can do dynamic programming with only two rows. We declare the matrix as a[2][X.length].
- When processing row i, a[i%2] is used as the current row while the other the previous.
- Of course, we need reset the elements of the array representing the current row at the beginning of each row process.

### **BIG INTEGER NUMBER & SUM**

Suppose one two big integer number

A = 1234858548957489574895784750894758

B = 3843248439048392493849038493489034



At first, 2 big numbers are placed two arrays, respectively. Then add their values based on indexing while addding carry to the next indexing value.

### Optimization: Big Integer Representation

- Supple length(X) = 10,000 and length(Y) = 100 then we need a matrix of size 10,000 \* 100 = 1,000,000.
- If each element of the matrix has array of size 100, then we need 100,000,000, a big number. It is also very time consuming to perform a huge number of add operations. You can get TLE.
- So we need to modified BIG INTEGER NUMBER & SUM calculation technique.
- Instead of storing one digit in one element of the array used for storing big numbers, we can store more digits. Suppose we store 4 digits. Then we need an array of 25 integers to represent 10<sup>100</sup> and computation time is reduced by factor 4.

Carry =13792/10000 sum =13792%10000