19I510 Design and Analysis of Algorithms

Exercise 4

String Matching

Learning Objective

1. Implement string matching algorithms and realize its execution for larger inputs.
2. Adopt efficient string matching algorithm for computationally intensive tasks.
3. Red is the founder of a new start-up game developer company. When Red developed his first game with his team, Red found the exact problem which he has learnt back in his undergraduate study, the string-matching problem. However, being an ignorant person, Red did not pay much attention on this subject and managed to barely pass the exam. Red delegated this problem to one of his new programmers who is also a fresh-graduate, with the hope that this new guy still remembers the linear time-complexity solution for this problem.

Unfortunately, instead of implementing a (correct) string matching algorithm, this new guy implemented a wrong one:

1. Let P be the pattern and S be the string.
2. If |P| > |S|, then output "NO" and terminate.
3. If P is a prefix of S, then output "YES" and terminate; otherwise
4. Let x be the smallest index where Px ≠ Sx,
5. Update S as suffix of S starting from index x + 1, then back to step 2.

Knowing that his solution is linear time-complexity, this new guy is confident that this solution works. However, of course you, being a competitive programmer, realize that this solution is simply wrong.

For example, let P = "ABABC" and S = "ABABABCABA".

|  |  |  |
| --- | --- | --- |
| First round: | S: ABABABCABA  P: **ABABC** | P is not a prefix of S and P4 ≠ S4 (x = 4 in 0-based index), so update S as suffix of S starting from index 4 + 1 (= 5): ABABABCABA → BCABA. |
| Second round: | S: BCABA  P: **A**BABC | P is not a prefix of S and P0 ≠ S0 (x = 0 in 0-based index), so update S as suffix of S starting from index 0 + 1 (= 1): BCABA → CABA. |
| Third round: | S: CABA  P: ABABC | |S| is lower than |P| (4 < 5), so output "NO" and terminate. |

Therefore, this algorithm will produce "NO" output for P = ABABC and S = ABABABCABA, even though we can find P in S: AB(ABABC)ABA.

You want to analyze the damages caused by this algorithm, so, as the first step, you should reproduce this algorithm. Given a pattern P and a string S, output whether P exists in S according to the aforementioned algorithm.

**Input Format**It contains two string P and S separated by a single space denoting the pattern and the string, respectively. P and S consist of uppercase alphabetic characters only (A-Z) and have length between 1 and 20,000 characters.

**Output Format**For each case, output either YES or NO

**Sample Input**

ABABC ABABABCABA

**Sample Output**  
NO

1. Implement a linear time correct algorithm for finding the pattern in the given text.

**Input Format**It contains two string P and S separated by a single space denoting the pattern and the string, respectively. P and S consist of uppercase alphabetic characters only (A-Z) and have length between 1 and 20,000 characters.

**Output Format**For each case, output either YES or NO

**Sample Input**

ABABC ABABABCABA

**Sample Output**

NO

1. Aragorn has started hunting creature Gollum. After several days following his footprints, he has arrived to the Dead Marshes. He has a map of the marshes, that can be viewed as an M1 \* M2 matrix containing lowercase letters form English alphabet (i.e. letters from ‘a’ to ‘z’).

Being a skilled ranger, Aragorn has been able to fully characterize Gollum preferred place (if you are interested, you should know that it must be dark, wet, creepy and full of fishes!). It can be described as an N1 \* N2 matrix containing lowercase letters form English alphabet.

Your task is simple: write a program that, given Gollum’s preferred place description and Aragorn’s map, output all possible locations of the creature.

Hint: Use a linear time sorting algorithm for every row in the Aragorn’s map to find the presence of first row of Gollum preferred place

**Input Format**The first line of input consists of N1 and N2

Lines 2 to N1+1 contains the string of N2 characters

The next line contains integers M1 and M2

The following M1+1 lines contain string of M2 characters

1 ≤ N1, N2 ≤ 300

1 ≤ M1 \* M2 ≤ 2000

N1 ≤ M1

N2 ≤ M2

**Output Format**On each line print the upper-left corner of all places that match Gollum’s preferred place description on the form “x y” without the quotes, where x stands for the row and y for the column. They should be lexicographically sorted, i.e. imagine them as an ordered pair. Then (x1,y1) < (x2,y2) if and only if x1 < x2 or, if they are equal, y1 < y2.

**Sample Input**

3 3

aba

bab

aba

7 6

aababa

ababab

bababa

ababab

ababab

bababa

ababab

**Sample Output**

1 2

1 4

2 1

2 3

5 1

5 3