5 Longest common substring

Problem Introduction

In the longest common substring problem one is given two strings s and t and the goal is to find a string w of maximal length that is a substring of both s and t. This is a natural measure of similarity between two strings. The problem has applications in text comparison and compression as well as in bioinformatics.

The problem can be seen as a special case of the edit distance problem (where only insertions and deletions are allowed). Hence, it can be solved in time $O(|s| \cdot |t|)$ using dynamic programming. Later in this specialization, we will learn highly non-trivial data structures for solving this problem in linear time $O(|s| + |t|)$. In this problem, your goal is to use hashing to solve it in almost linear time.

Problem Description

Input Format. Every line of the input contains two strings s and t consisting of lower case Latin letters.

Constraints. The total length of all s's as well as the total length of all t's does not exceed 100 000.

Output Format. For each pair of strings s and t_i, find its longest common substring and specify it by outputting three integers: its starting position in s, its starting position in t (both 0-based), and its length. More formally, output integers $0 \le i < |s|$, $0 \le j < |t|$, and $l \ge 0$ such that $s_i s_{i+1} \cdots s_{i+l-1} = t_j t_{j+1} \cdots t_{j+l-1}$ and l is maximal. (As usual, if there are many such triples with maximal l, output any of them.)

Time Limits. `C`: 2 sec, `C++`: 2 sec, `Java`: 5 sec, `Python`: 15 sec. `C#`: 3 sec, `Haskell`: 4 sec, `JavaScript`: 10 sec, `Ruby`: 10 sec, `Scala`: 10 sec.

Memory Limit. 512MB.

Sample 1.

Input:

```
cool toolbox
aaa bb
aabaa babbaab
```

Output:

```
1 1 3
0 1 0
0 4 3
```

Explanation:
The longest common substring of the first pair of strings is `ool`, it starts at the first position in `toolbox` and at the first position in `cool`. The strings from the second line do not share any non-empty common substrings (in this case, $l = 0$ and one may output any indices i and j). Finally, the last two strings share a substring `aab` that has length 3 and starts at position 0 in the first string and at position 4 in the second one. Note that for this pair of string one may output `2 3 3` as well.

What to Do

For every pair of strings s and t, use binary search to find the length of the longest common substring. To check whether two strings have a common substring of length k,

- precompute hash values of all substrings of length k of s and t;
- make sure to use a few hash functions (but not just one) to reduce the probability of a collision;

- store hash values of all substrings of length k of the string s in a hash table; then, go through all substrings of length k of the string t and check whether the hash value of this substring is present in the hash table.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.