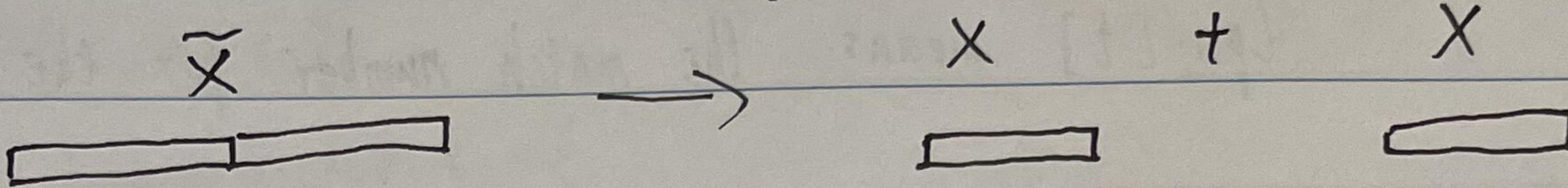


Question 2

First, create a new \tilde{X} by linking 2 X



Then we use \tilde{X} as the t sequence in Z-algorithm, and use Y as the p sequence.

input : $t \quad \tilde{X} \quad \text{---} \quad |p| < |t|$
 $P \quad Y \quad \text{---}$

Finally, use Z-algorithm to find matching pattern.

If the sequence Y is in sequence \tilde{X} , it means X and Y come from an identical ~~se~~ circular sequence. Otherwise X and Y come from different circular sequences.

Question 3

$S = \text{TCATCATGATGATCATCT}$ Length = 18

$P = \text{ATCATCT}$ Length = 7

$lps = [0, 0, 0, 1, 2, 3, 0]$

$lps' = [0, 0, 0, 0, 0, 3, 0]$

KMP Search:

Step 1: $\text{TCA} \dots$

$\text{ATC} \dots$

$\times \ lps[1] = 0 \quad \text{skip } 1-0=1 \text{ bytes}$

Step 2: $\text{TCA} \dots$

$\text{AT} \dots$

$\times \ lps[2] = 0 \quad \text{skip } 1-0=1 \text{ byte}$

Step 3: $\text{TCATCATGATGA} \dots$

ATCATCT

$\times \ lps[3] = 3 \quad \text{skip } 6-3=3 \text{ bytes}$

Step 4: $\text{TCATCATGATGA} \dots$

$\text{ATCATCT} \dots$

$\times \ lps[4] = 0 \quad \text{skip } 3-0=3 \text{ bytes}$

Step 5: $\text{TCATCATGATGA} \dots$

$\text{ATCA} \dots$

$\times \ lps[5] = 0 \quad \text{skip } 3-0=3 \text{ bytes}$

Step 6: $\text{TCATCATGATGA} \text{TCATCT}$

$\text{ATCATCT} \underline{\text{match}}$

$lps[4]$ and $lps[5]$ are not used, so there is no difference when using lps' instead lps .

However, I think the claim is valid.

For example:

$S = TCA\ TCA\ CACAT\ CATCT$

$P = ATCATCT$

$lps = [0, 0, 0, 1, 2, 3, 0]$

$lps' = [0, 0, 0, \cancel{1}, 0, 3, 0]$

using lps

step 3: $TCA\ TCA\ CACAT\ CATCT$

$ATCATCT$

$\times lps[5] = 2$

skip $5-2=3$ bytes

using lps'

$TCA\ TCA\ CACAT\ CATCT$

$ATCATCT$

$\times lps'[5] = 0$

skip $5-0=5$ bytes

step 4: $TCA\ TCA\ CACAT\ CATCT$

$ATCATCT$

$\times lps[2] = 0$

skip $2-0=2$ bytes

$ATCATCT$

match

step 5: $TCA\ TCA\ CACAT\ CT$

$ATCAT\ CT$

match

no need 1 more step

At this situation, using lps will increase 1 step compared to using lps' , which means lps' is more efficient than lps .

In my opinion, if matching stops at position i and $lps[i]$ is not '0', the next step will not get right match. So if we want to accelerate the process, we need to reduce non-zero value in lps . And in lps , later non-zero value will cover the front position, which means the claim in the Question 3 will not result in missing right match.

Question 4

KMP Algorithm

lps[i] means the match number for the string ending with i

Z Algorithm :

`z[i]` means the match number for the string beginning with i

The diagram illustrates the KMP string matching algorithm. It shows three horizontal lines representing the input string, pattern string, and a step counter.

- Input String:** The top line contains the string "PQRST".
- Pattern String:** The middle line contains the string "PQRST".
- Step Counter:** The bottom line contains the string "match".

Arrows indicate the current character being compared:

- A vertical arrow labeled 'P' points to the first 'P' in the pattern string.
- A horizontal arrow labeled '1' points to the first 'P' in the input string.
- A horizontal arrow labeled '2' points to the second 'Q' in the input string.
- A horizontal arrow labeled '3' points to the third 'R' in the input string.
- A horizontal arrow labeled '4' points to the fourth 'S' in the input string.
- A horizontal arrow labeled '5' points to the fifth 'T' in the input string.

The diagram illustrates the string matching process between a pattern P and a text Z . The text Z is shown as a horizontal line with several segments labeled i , $i+1$, and $i+Z[i]-1$. The pattern P is shown below Z with a bracket indicating its length. A vertical arrow points from the label $Z[i]$ to the character under the index i in the text Z .

$Z[i]$ means the length of matching string

$$\text{so } lps[i+Z[i]-1] = Z[i]$$

pseudo code: input: string S , Zarray

Output: Lps'

$$1) \quad Lps' = \underbrace{0, 0, \dots, 0}_{\text{length of } s}$$

2) for i in range(1, length(s]):

if $Z[t] > 0$:

$$lps[i + z[i] - 1] = z[i]$$