

KMP Algorithm

text → "abccabdyxdae"

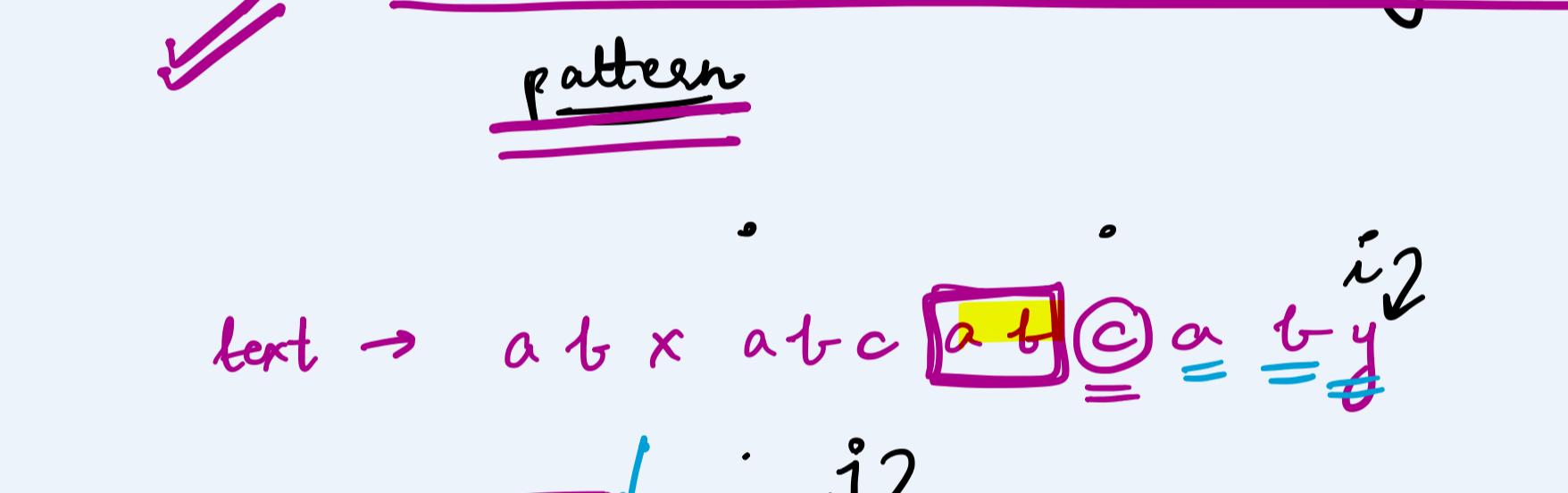
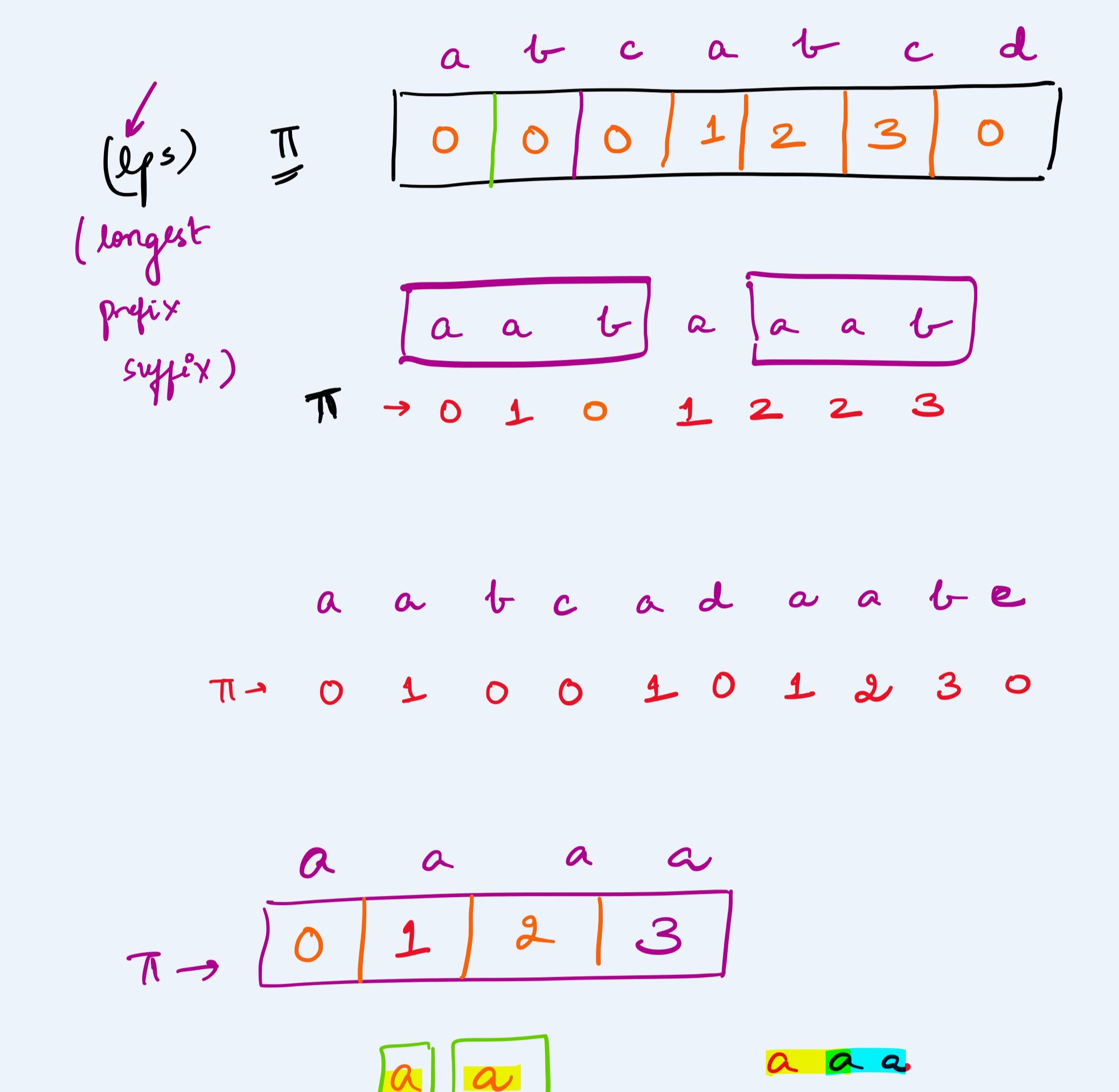
pat → "cdyxda"

Find the occurrence of pattern in the text.

→ For the given pattern, we will calculate its prefix function.

You have a string of length  $n$ . The prefix function for the string is defined as an array  $\pi$  of length  $n$ , where  $\pi[i]$  is the length of proper prefix of the substring  $s[0, \dots, i]$  which is also a suffix of the substring  $s[0, \dots, i]$ .

Proper prefix: If a string is a prefix that is not equal to the string itself.



text → abc x abc dat x abc da b c d a t y

pat → abc x abc dat y      lps → 0      lps → 3

① First calculate the  $\pi$  array for the pattern

text → ab x abc a b @ a = b y

pat → a b o o o y      lps → 0

$\pi[j-1] = 0$

 $\pi[0 \rightarrow j-1]$  $\pi[0 \rightarrow 1]$ 

KMP (text, pat) :

lps[0]

i = 0;

j = 0;

while ( $i < \text{text.length}$  &  $j < \text{pat.length}$ ) {if ( $\text{text}[i] == \text{pat}[j]$ ) {

i++;

j++;

} else {

if ( $j == 0$ ) {

i++;

} else {

 $j = \text{lps}[j-1]$ 

}

}

if ( $j == \text{pat.length}$ ) {

return true;

}

else {

lps[i] = -1;

i++;

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}