KMP Algo

(Nuth)

Knuth - Morris - Pratt Algo

Naive was $(n-m+1)\,m$

checking ahead again & again

=

Now KMP Goal : Take advantage of successful compari

Prefix & Suffix.

we find this in the pattern

So a.) ds gw a ds 2

prefix suffix

⇒ pattern a b c d a b c

we need subset of pattern as

prefix & suffix

(2)

for a b c d a b c

Prefix examples a, or ab, ~~cet~~ abc

Suffix example c, bc, abc, dabc

⇒ Goal of KMP? find prefix
Same as suffix in the pattern?

So for

abc d abc

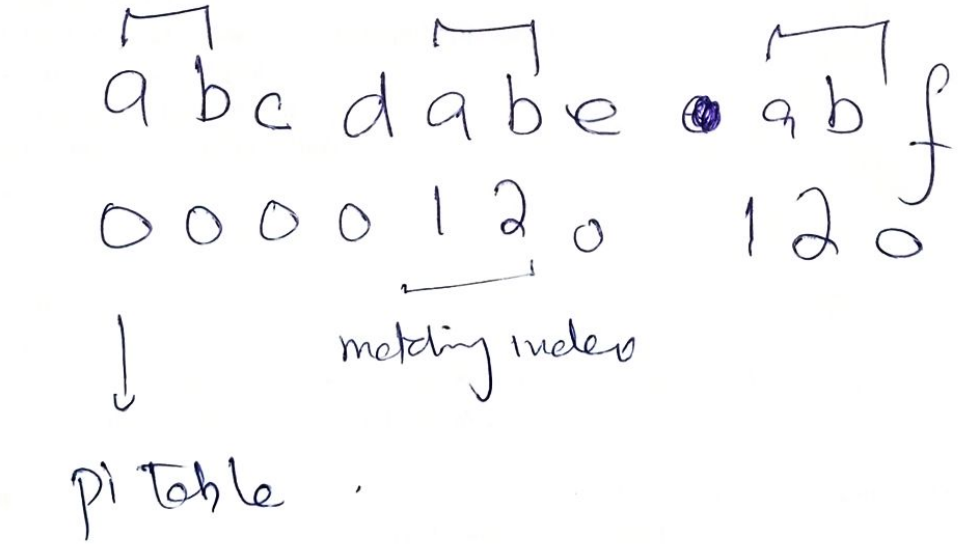
prefix Suffix

This observation will be used in KMP to avoid comparisons

⇒ In KMP, we try to identify if beginning of a string appears again?

To find this, we need to make a pi table (π table) also called longest prefix which is same as some suffix (LPS)

e.g.



pi Table.

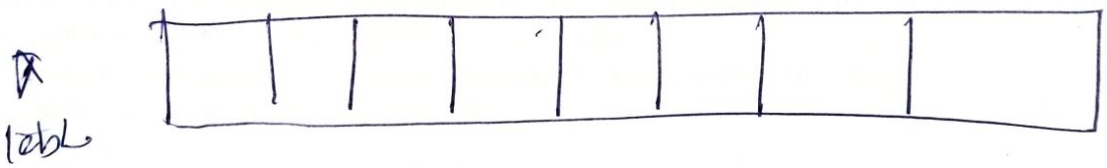
another eg

a b c d e a b f a b c
0 0 0 0 0 1 2 0 1 2 3

How it is made

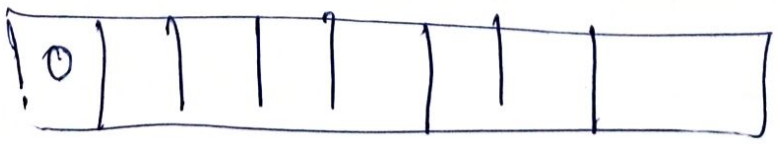
working

a b c d a b c a
0 1 2 3 4 5 6 7



first is always zero, we need j & i

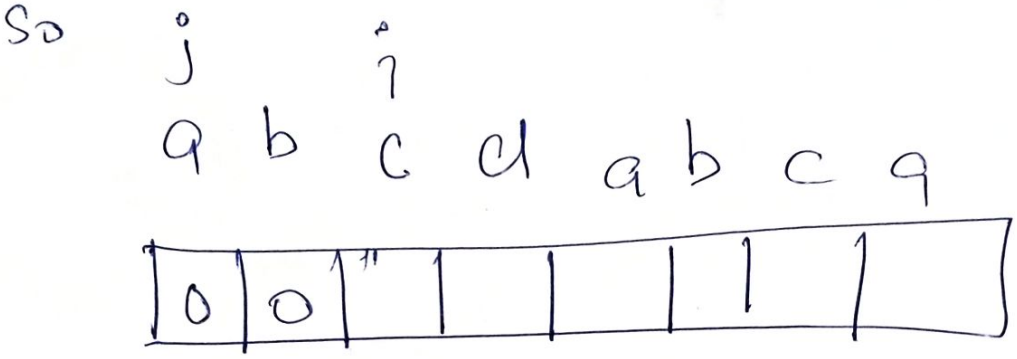
a b c d a b c a
0



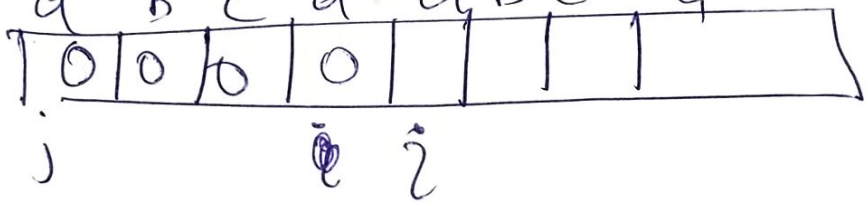
if $i \neq j$

put zero & increment i

So

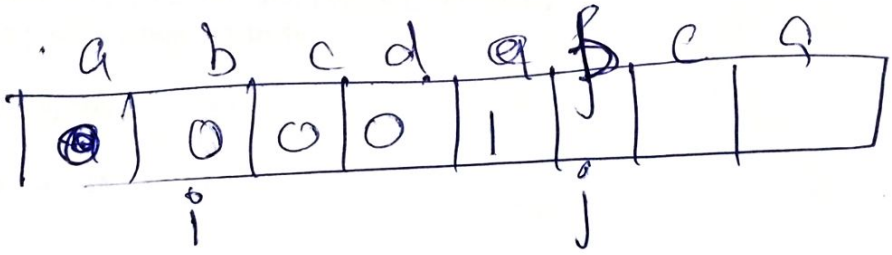


as $i \neq j$ so zero & increment i

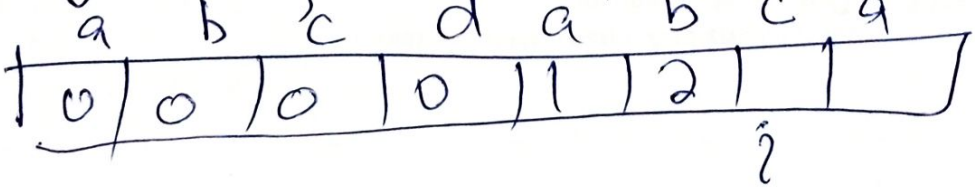


★ Now $i = j$ so value of $j+1$

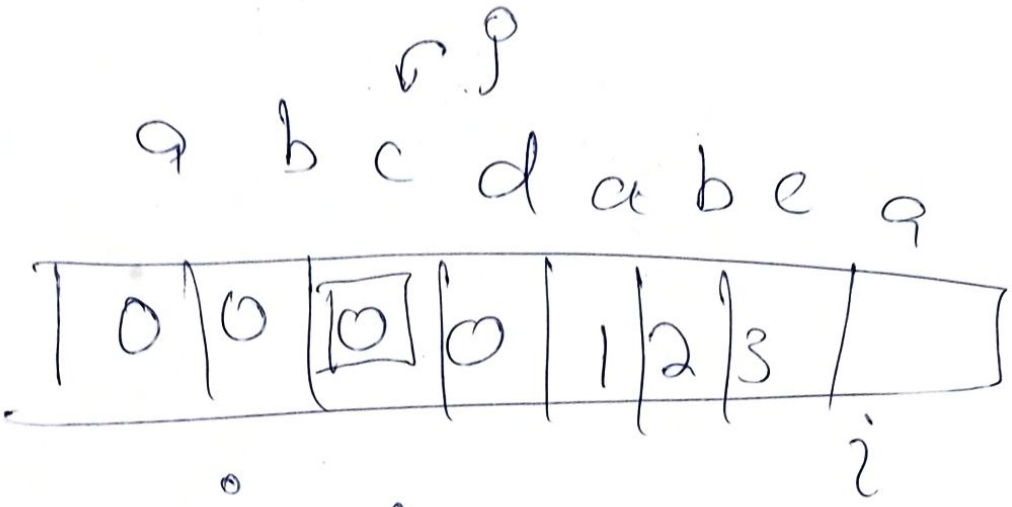
(so $0+1=1$, $i++$ & $j++$



again so $1+1=2$, $i++$, $j++$



Now



Time O(n) & Space = O(m

An exemple

a a b a a b a a a

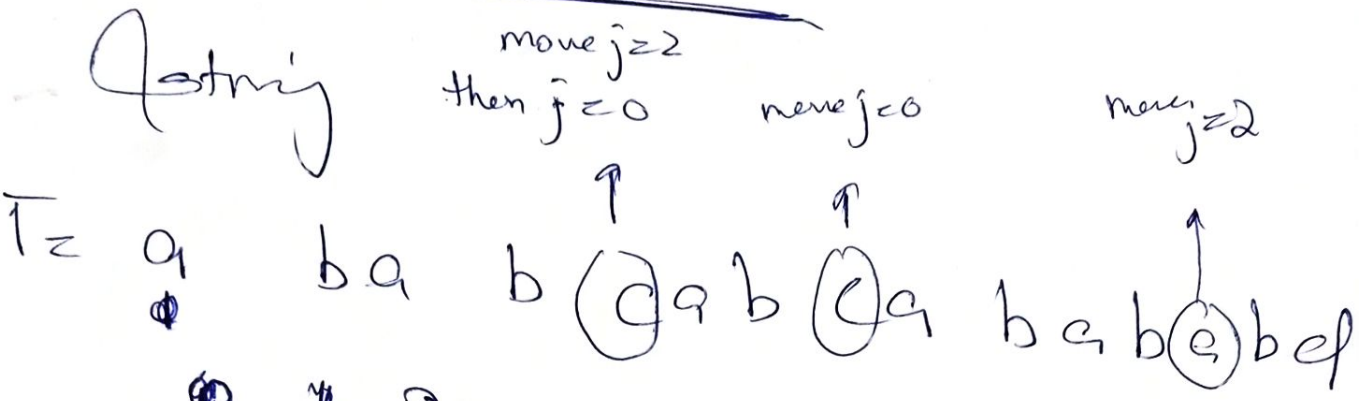
j jumps 2.
j jumps 1

S_0

a a b a a b a a a

0	1	0	1	2	3	4	5	2

How KMP works



Start j from zero

So check.

if i = j

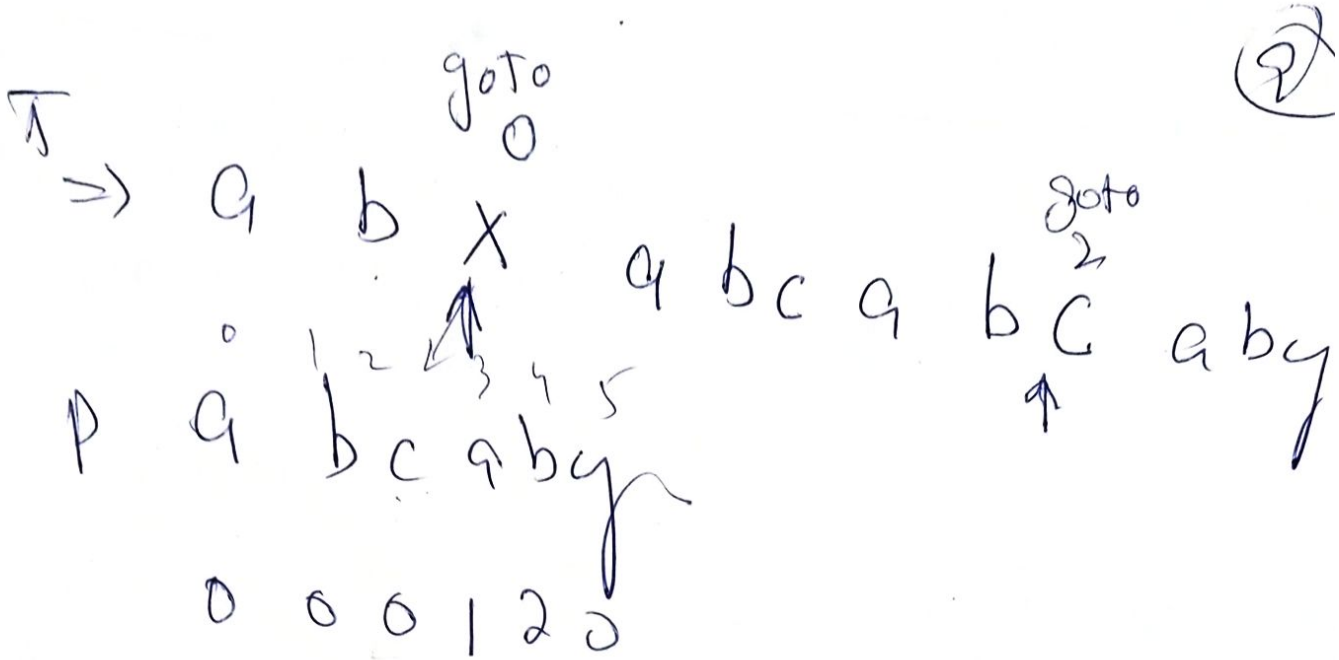
if match, move i++ & j++

if mismatch, move j to j-1 the value

So ~~m[2]~~ j = 2

if again mismatch, move j backward if j = 0, i++.

Complexity O(n)


goto
0
goto
2
a b X a b c a b C a b y
0 1 2 3 4 5
p a b c a b y
0 0 0 1 2 3