KMP

It's a string matching algorithm. It tries to find a given pattern in a string.

TC $O(n)$ n = length of string
SC $O(m)$ m = " "   pattern.

It uses longest prefix suffix (lps or π) of the pattern to find in string.

str: abcmnab
prefix: a, ab, abc, abcm, abcmn, abcmna, abcmnab
suffix: b, ab, nab, mnab, cmnab, bcmnab, abcmnab

LPS (π)

pattern: a b c d a b e a b f
            ⟨0 0 0 0⟩ 1 2 0 1 2 0 → again 0 because new pattern

zero occurences yet ↗   now pattern breaks new char so 0

a came earlier so 1
∴ a = 1   b = 1+1 = 2

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

pattern: a a b c a d a a b c
            0 1 0 0 1 0 1 2 3 4

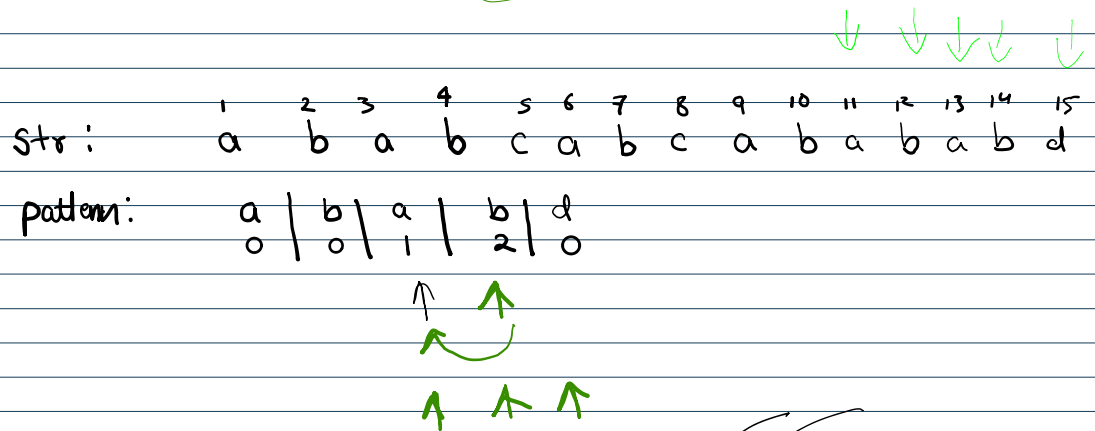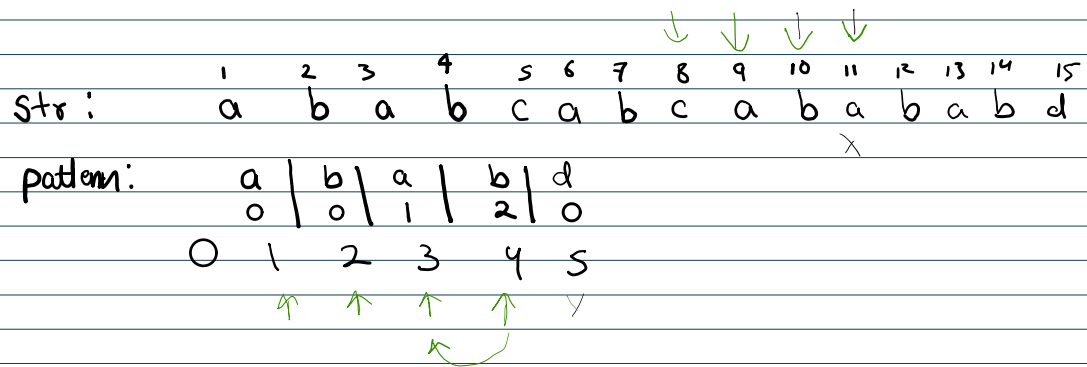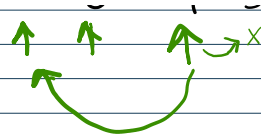pattern: a a a a b a a c d
            0 1 2 3 0 1 2 0 0

pattern: a b a b c a b c a b a b a b d
            0 0 1 2 0 1 2 0 1 2 3 4 3 4 0

            ↓   ↓   ↓   ↓   ↓
i →   1   2   3   4   5   6   7   8   9   10 11 12 13 14 15
str: a   b   a   b   c   a   b   c   a   b   a   b   a   b   d
                                ✗

pattern: a | b | a | b | d
            0 | 0 | 1 | 2 | 0
            0   1   2   3   4 ✗ 5
j+1 →   ↑   ↑   ↑   ↑

:

            ⇓⇓   ↓ ↓   ↓   ✗
            1   2   3   4   5   6   7   8   9   10 11 12 13 14 15
str: a   b   a   b   c   a   b   c   a   b   a   b   a   b   d

pattern: a | b | a | b | d
            0 | 0 | 1 | 2 | 0
            0   1   2   3   4   5
                        ↑   ↑   ↑ ✗
                        ↑   ↑

↑ ↑ ↑ ↪x

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Str:	a	b	a	b	c	a	b	c	a	b	a	b	a	b	d

pattern:

a	b	a	b	d
0	0	1	2	0

0 1 2 3 4 5

↑ ↑ ↑ ↑

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Str:	a	b	a	b	c	a	b	c	a	b	a	b	a	b	d

pattern:

a	b	a	b	d
0	0	1	2	0

↑ ↑

↑ ↑ ↑

✓

notes:

*   works on prefix and suffix.