KMP Advanced Question

Naive Pattern Matching $O(m*n)$
↓
KMP $O(m+n)$

① Failure Function construction

Pattern : "ABABCABAB"

A	B	A	B	C	A	B	A	B
-1	0	0	1	2	0	1	2	3

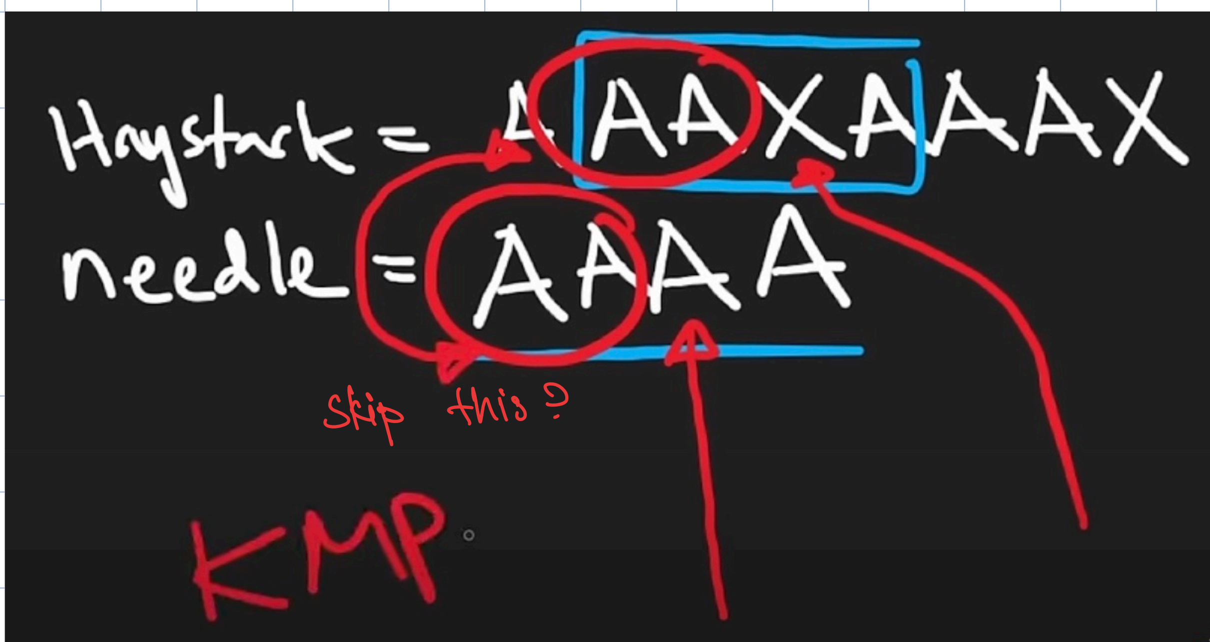
✓

② 2025 Sample Exam

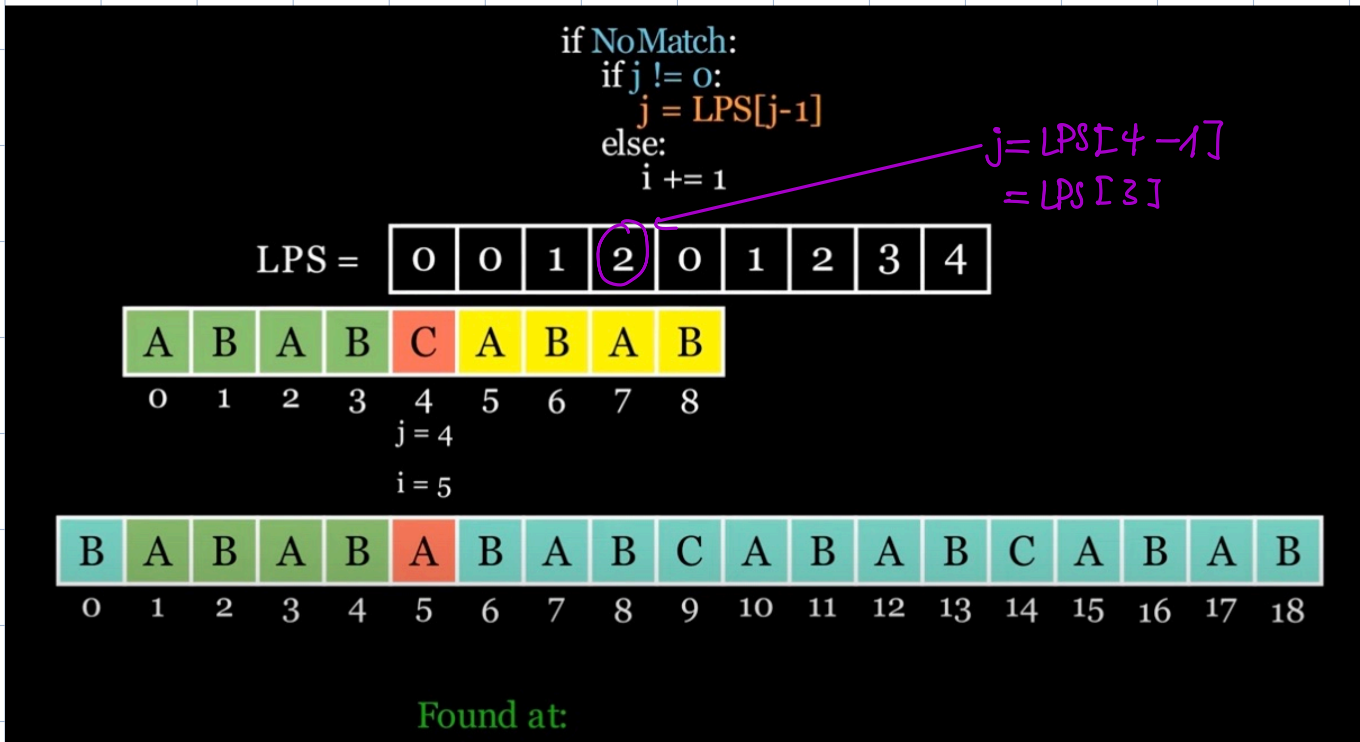
a	p	p	l	e	'	s	␣	a	p	p	␣	s	t	o	r	e
-1	0	0	0	0	0	0	0	0	1	2	3	0	0	0	0	0

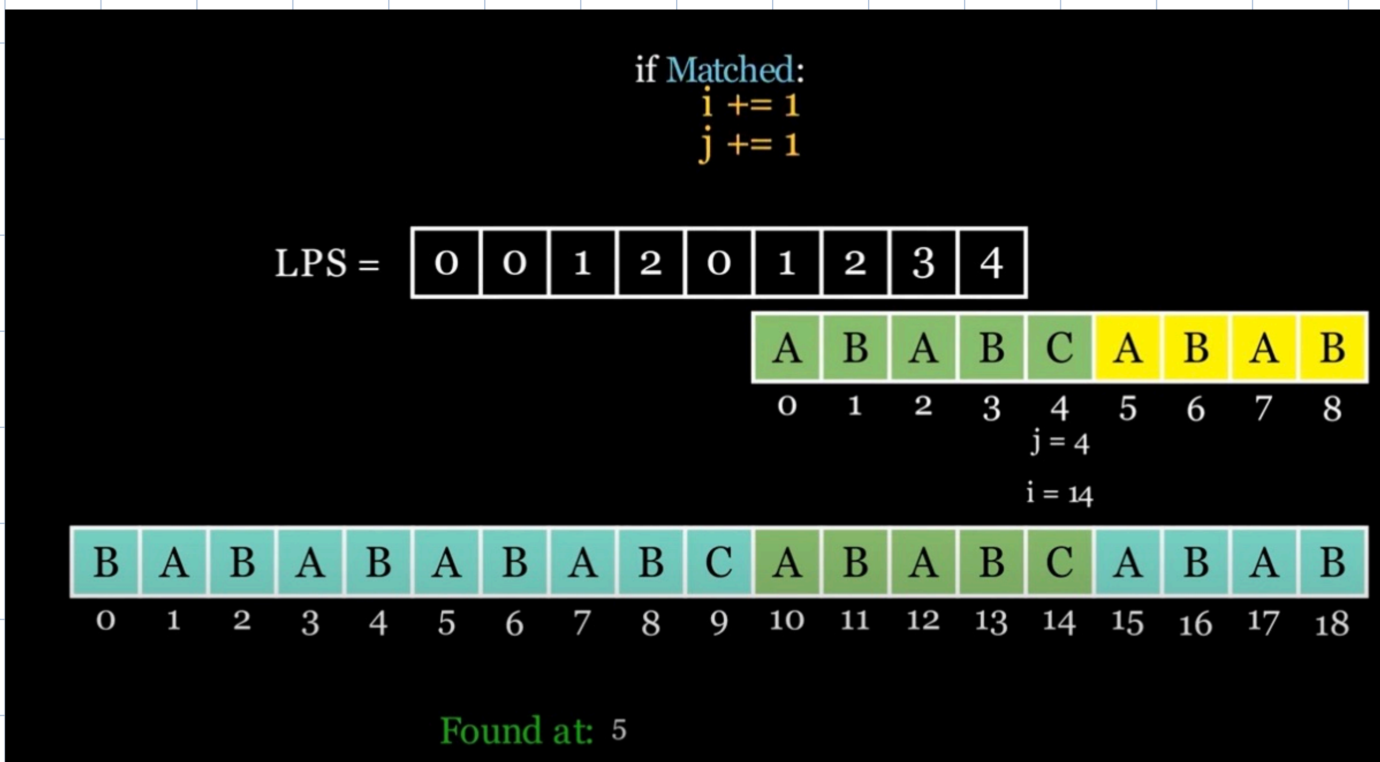
✓

KMP algorithm recap:



"AA": longest prefix suffix (LPS) array ~ Failure function


if NoMatch:
if j != 0:
j = LPS[j-1]
else:
i += 1
j= LPS[4-1]
= LPS[3]
LPS = 0 0 1 2 0 1 2 3 4
A B A B C A B A B
0 1 2 3 4 5 6 7 8
j = 4
i = 5
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
Found at:


if Matched:
i += 1
j += 1
LPS = 0 0 1 2 0 1 2 3 4
A B A B C A B A B
0 1 2 3 4 5 6 7 8
j = 4
i = 14
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
Found at: 5

so you fall back to the length of the longest proper prefix-suffix of pat[0..j−1]. In your COMP10002 “failure” style you have already built

```r
F[0] = −1;
F[1] =  0;
for (i = 2; i <= m; i++)
    F[i] = lps[i−1];
```

(which is simply lps shifted by one cell, with a sentinel at F[0]). That means the exact analogue of `j = lps[j−1]` becomes

```c
while (j >= 0 && text[i] != pat[j])
    j = F[j];
```

because for any j > 0, F[j] = lps[j-1], and when j hits zero you get F[0] = −1. The −1 then forces both i++ and j++ (to zero) on the next iteration, so you make progress just as you would with the separate `j > 0` check in the lps version.

Q3: KMP Pattern Matching Trace

Text : A B A B A B C A B A B C A B A B

Pattern: A B A B C A B A B

(a) Construct the failure function F

(b) Trace through the KMP algorithm execution step by step

(a)

A	B	A	B	C	A	B	A	B
−1	0	0	1	2	0	1	2	3
0	1	2	3	4	5	6	7	8

✓

{ Match at 2 = 11 − 9 }

(b) * Trace through the KMP algorithm:

i

Text :	A	B	A	B	A	B	C	A	B	A	B	C	A	B	A	B
Pattern :	A	B	A	B	C	A	B	A	B	j						
Index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Step 1: $i=0, j=0$: $\boxed{A == A \checkmark}$ $\therefore$ $i=1, j=1$

Step 2: $i=1, j=1$: $\boxed{B == B \checkmark}$ $\therefore$ $i=2, j=2$

Step 3: $i=2, j=2$ $\therefore$ $\boxed{A == A \checkmark}$ $\therefore$ $i=3, j=3$

Step 4: $i=3, j=3$ $\therefore$ $\boxed{B == B \checkmark}$ $\therefore$ $i=4, j=4$

Step 5: $i=4, j=4$ $\therefore$ $\boxed{A \,!= C \times}$ $\therefore$ Use $F[4]=2$, $j=2$

$F[i] = F[4] = 2$

No match: i const, $j = F[i]$

$\therefore$ $i=4, j=2$

Step 6: $i=4, j=2$ $\therefore$ $\boxed{A == A \checkmark}$ $\therefore$ $i=5, j=3$

Step 7: $i=5, j=3$ $\therefore$ $\boxed{B == B \checkmark}$ $\therefore$ $i=6, j=4$

Step 8: $i=6, j=4$ $\therefore$ $\boxed{C == C \checkmark}$ $\therefore$ $i=7, j=5$

Step 9: $i=7, j=5$ $\therefore$ $\boxed{A == A \checkmark}$ $\therefore$ $i=8, j=6$

Step 10: $i=8, j=6$ $\therefore$

⋮

step 12: $i = 10, j = 8$: $\boxed{B == B}$ ✓ : $i = 11, j = 9$

$\Rightarrow$ Match at position 2 $(i - j = 11 - 9)$