Knuth-Morris-Pratt (KMP) Algorithm

Concepts

String: ABCDE

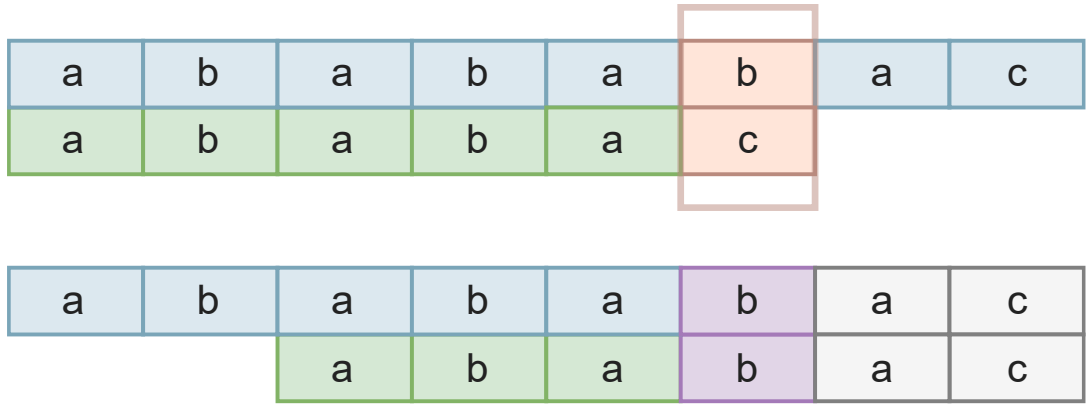
Proper prefixes: <NULL>, A, AB, ABC, ABCD

Proper suffixes: <NULL>, E, DE, CDE, BCDE

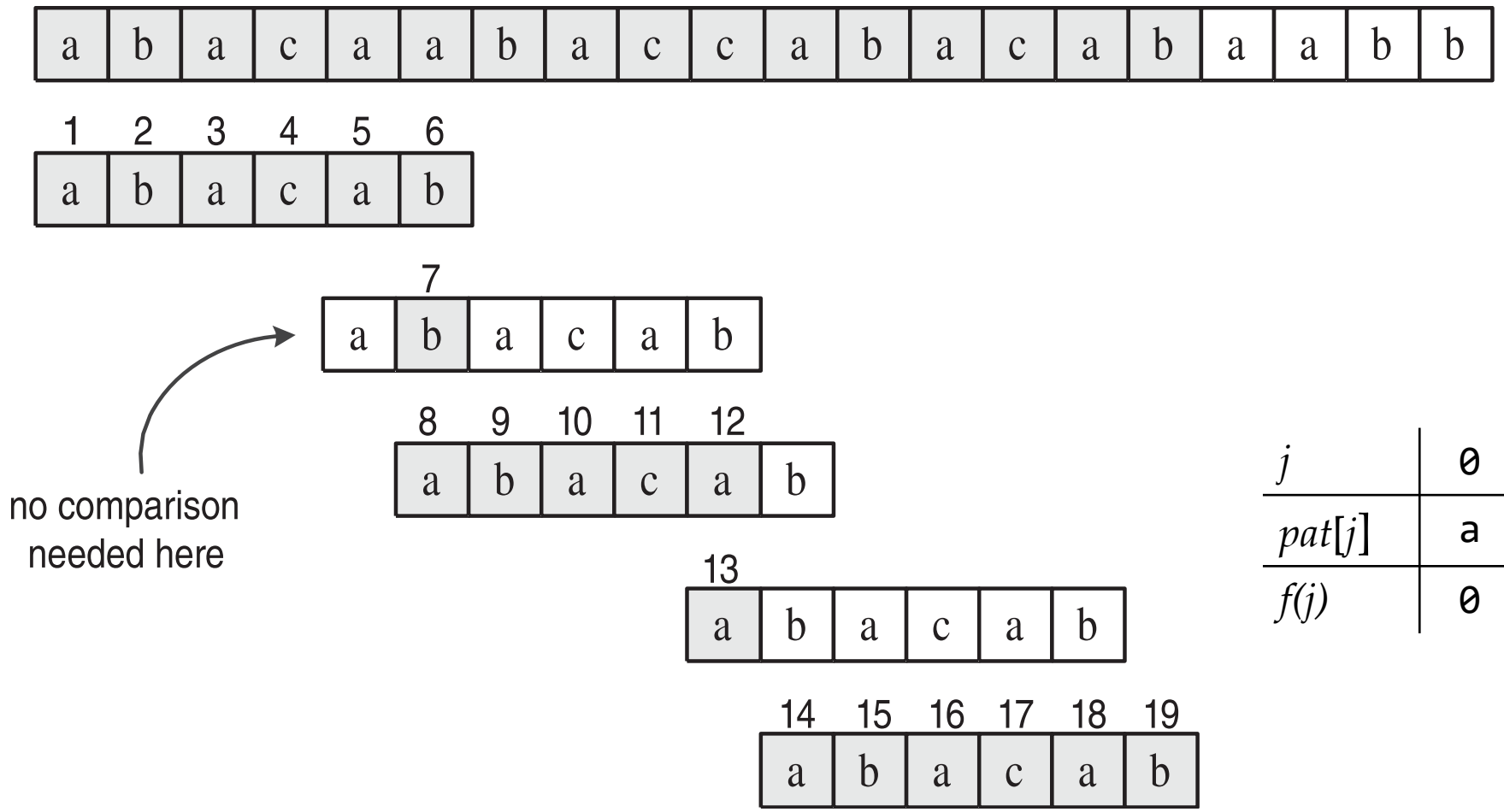
Substring of interest: Largest common substring that is both prefix & suffix

Failure function: The length of the longest prefix of **P** that is a suffix of P[1..j] and f(O) = O

Intent: Upon mismatch, shift the string so that prefix replaces suffix



Working example of KMP search using failure function



j	0	1	2	3	4	5
$pat[j]$	a	b	a	c	a	b
$f(j)$	0	0	1	0	1	2

Another example of KMP search

Step 1		A	B	C	**A**	B	C	D	A	B	A	B	C	D	A	B	C	D	A	B	D	E
		A	B	C	**D**	A	B	D														
Step 2		A	B	C	**A**	**B**	C	D	**A**	**B**	**A**	B	C	D	A	B	C	D	A	B	D	E
					A	**B**	C	D	**A**	**B**	**D**											
Step 3		A	B	C	A	B	C	D	A	B	**A**	B	C	D	A	B	C	D	A	B	D	E
									A	B	**C**	D	A	B	D							
Step 4		A	B	C	A	B	C	D	A	B	**A**	**B**	C	D	**A**	**B**	**C**	D	A	B	D	E
											A	**B**	C	D	**A**	**B**	**D**					
Step 5		A	B	C	A	B	C	D	A	B	A	B	C	D	A	B	C	D	A	B	D	E
															A	B	C	D	A	B	D	

j	0	1	2	3	4	5	6
$pat[j]$	A	B	C	D	A	B	D
$f(j)$	0	0	0	0	1	2	0

Reference: https://blog.devgenius.io/string-matching-kmp-ccad1ffbeb11

Complexities?