Arrays, Strings & Linked Lists Lecture 5

Saturday, 27 July 2024 3:04 PM

String Matching

pat in txt

$|pat| = m$ $|txt| = n$

Naive : $O(m \cdot n)$

Rabin-Karp : $O(m+n)$

KMP algorithm

Knuth-Morris-Pratt

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Txt :	a	b	a	b	c	a	b	c	a	b	a	b	a	b	d
									✓	✓	✓	✓	✗		

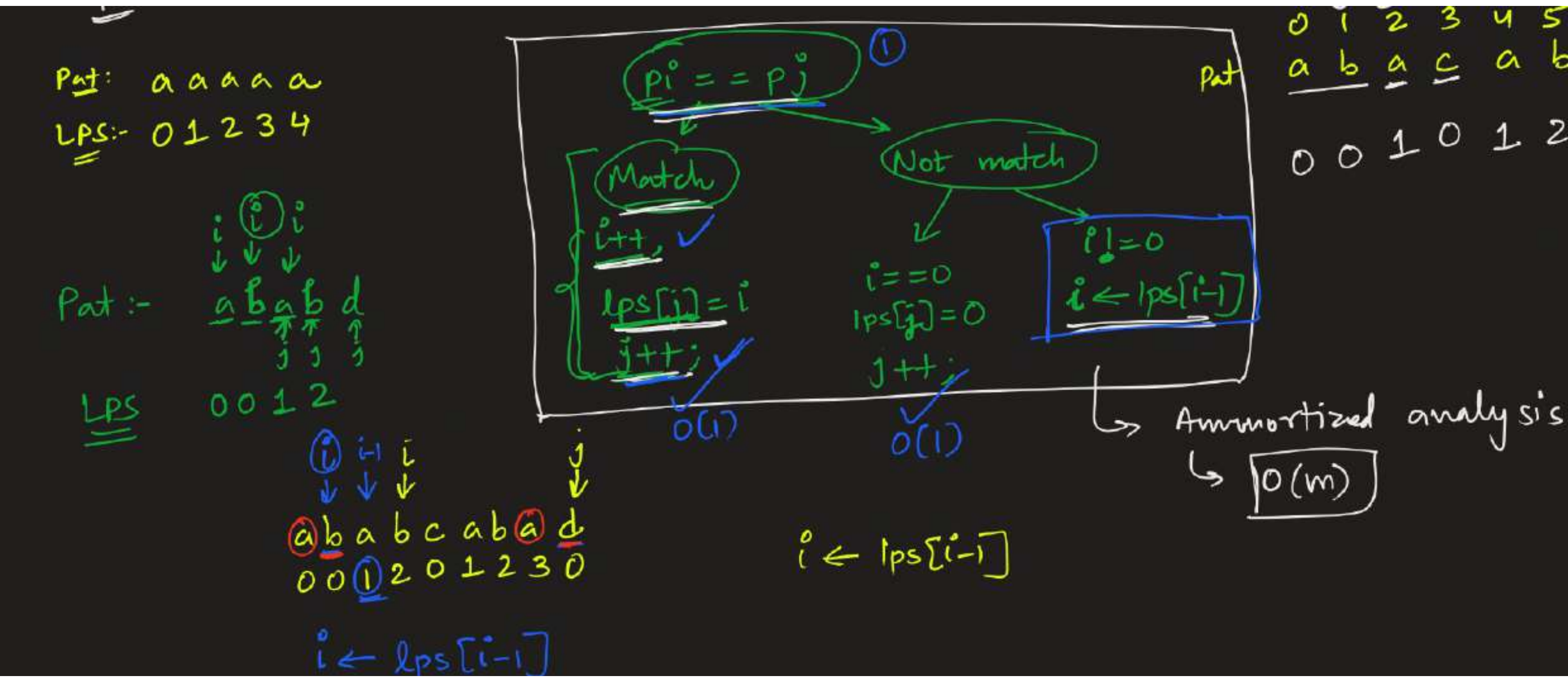
$|Txt| = n = 15$

	0	1	2	3	4
Pat :	a	b	a	b	d
	✓	✓	✓	✓	✗
			↑	↑	
LPS / Pi	0	0	1	2	0

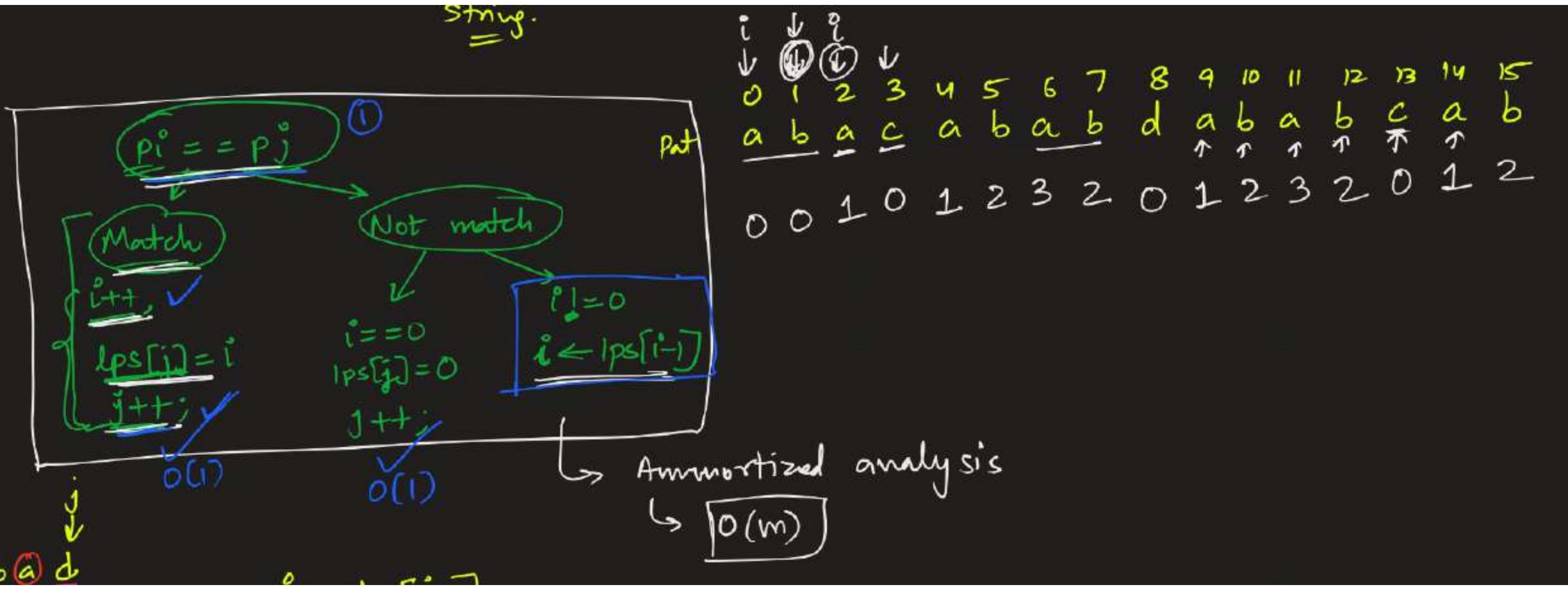
$|Pat| = m = 5$

$\underline{ab}\,\underline{ab}$ → Find pre and suffix such that they are not equal to the string.

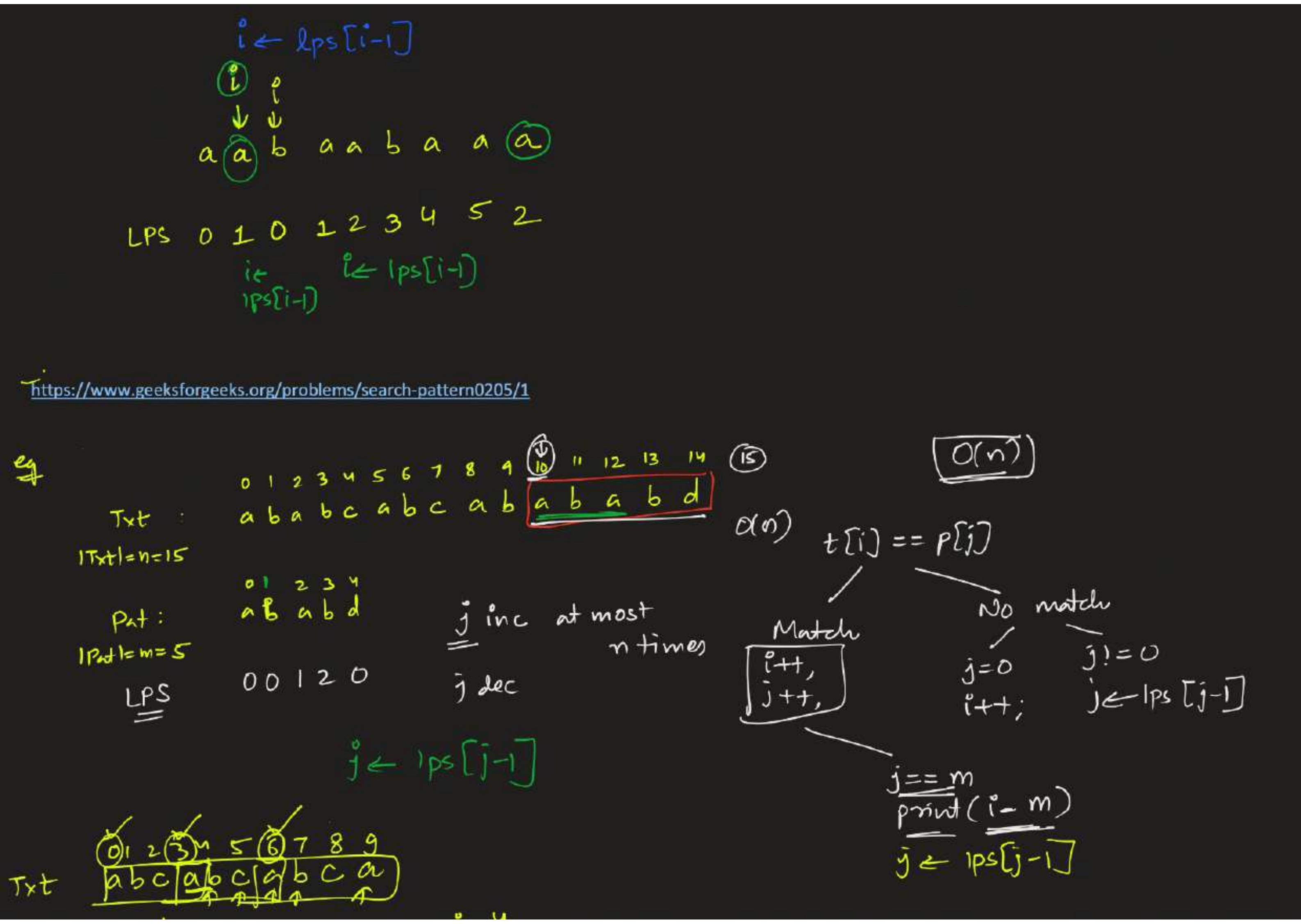
$\underline{a}bab\underline{d}$



Pat: a a a a a
LPS:- 0 1 2 3 4
Pat :- a b a b d
LPS 0 0 1 2
a b a b c a b a d
0 0 1 2 0 1 2 3 0
i ← lps[i-1]
pi == pj
Match
i++,
lps[j] = i
j++;
O(1)
Not match
i == 0
lps[j] = 0
j++;
O(1)
i != 0
i ← lps[i-1]
Ammortized analysis
O(m)
i ← lps[i-1]
Pat
0 1 2 3 4 5
a b a c a b
0 0 1 0 1 2



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	b	a	c	a	b	a	b	d	a	b	a	b	c	a	b
0	0	1	0	1	2	3	2	0	1	2	3	2	0	1	2



$i \leftarrow lps[i-1]$

a a b a a b a a a

LPS 0 1 0 1 2 3 4 5 2

$i \leftarrow lps[i-1]$ $\quad$ $i \leftarrow lps[i-1]$

https://www.geeksforgeeks.org/problems/search-pattern0205/1

eg

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Txt :	a	b	a	b	c	a	b	c	a	b	a	b	a	b	d	

|Txt| = n = 15

	0	1	2	3	4
Pat :	a	b	a	b	d
LPS	0	0	1	2	0

|Pat| = m = 5

$O(n)$

$\boxed{O(n)}$

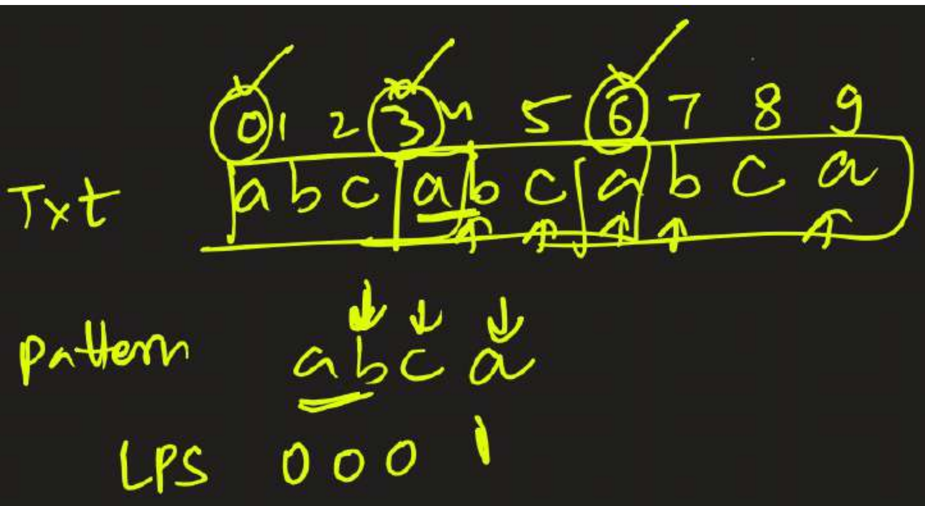
j inc at most n times

j dec

$j \leftarrow lps[j-1]$

$$\begin{tikzcd}
 & t[i] == p[j] \arrow[dl] \arrow[dr] & & \\
\text{Match: } \boxed{i++,\ j++,} \arrow[dr] & & \text{No match} \arrow[d] \arrow[r] & j != 0 \quad j \leftarrow lps[j-1] \\
 & j == m \quad print(i-m) \quad j \leftarrow lps[j-1] & j = 0 \quad i++; &
\end{tikzcd}$$

	0	1	2	3	4	5	6	7	8	9
Txt	a	b	c	a	b	c	a	b	c	a



$i = 4$

$j = 4$

T.C :- $\boxed{O(n+m)}$

```
def calc_lps(p):
    m = len(p)
    lps = [0]
    i = 0
    j = 1
    while j < m:
        if p[i] == p[j]:
            i += 1
            lps.append(i)
            j += 1
        else:
            if i == 0:
                lps.append(0)
                j += 1
            else:
                i = lps[i-1]
    return lps

class Solution:
    def search(self, pat, txt):
        lps = calc_lps(pat)
        # print(lps)
        ans = []
        i = 0
        j = 0
        n = len(txt)
        m = len(pat)
        while i < n:
            if txt[i] == pat[j]:
                i += 1
                j += 1
                if j == m:
                    ans.append(i-m+1)
                    j = lps[j-1]
            else:
                if j == 0:
                    i += 1
                else:
                    j = lps[j-1]
        return ans
```

TC :- $O(n+m)$

SC :- $O(m)$