CLASS-30

→ Stack, Queue, Dequeue

→ STL

→ Stack, Queue Implementation using Array

→ Stack, Queue using Queues, Stacks

→ Circular Queue

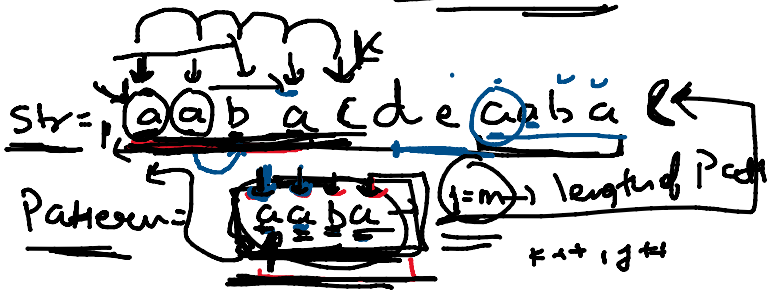
→ Question

Strings → STL → Substr erase insert

Pattern Matching Algo (OA + Interview)

Rabin Karp | Z Algorithm | KMP Algo

Pattern Matching



Brute Force

m + m + m + - - - - n

For (i=0; i<n; i++)

O(nm)

→ i=m index

for (i=0; ...)   O(n*m)

{ i^th index   index → substring

int k = i;   O(m)

int j = 0;

T.C = O(n*m)
Pat length

```
while ( k<n and j<m )
{ if ( str[k] == Pat[j] )
      { k++; j++;
      }
  else
      break;
}
if (j==m)
{ cout << " we found Pattern at :- " << k-m or i
}
}
```

break

k - m

Prefix , suffix

abcd → a, ab, abc, abcd
  ↳ d, cd, bcd, abcd

n length
n-1

Proper Prefix

LPS → longest common Proper Prefix and suffix

abab → " , a, (ab), aba
  ↳ b, (ab), bab, abab

} longest common Proper Prefix and suffix.

(2)

aba → "", a, ab] longest common Proper Prefix and
└→ a, ba, aba] Suffix (1)

lps array

abab

i=1

ab → "", a] 0
└→ b, ab

i=2 aba → a, ab } ⇒ 1
└ a, ba, aba

abab → a, ab, aba ⇒ 2
└→ b, ab, bab, abab

i=0 a → " " } 0 → lps
→ a

lps = 0

abab

aba

lps[] = {0, 0, 1, 2}

↓

abab longest comm = 2
suffix = 2

ababc

→ ab → "", a
└→ b, ab

aba
└→ a, ab
└→ a, ba, aba

abab → a, ab, aba
└→ b, ab, bab, abab

lps[] = {0, 0,

ababc → a, ab, aba, abab
└→ c, bc, abc, babc, ababc
└→ 0

aaaa

a a a a

i=0 'a' → 0

i=1 aa → 1 → 'a' → a, aa

i=2 aaa → a, aa → a, aa, aaa

i=3 aaaa → a, aa, aaa → a, aa, aaa, aaaa

↳ a, b, c, ab, bc, abc, babc, ab abc

↳ 0

(n) (n-1)

dps = (n)

dps →

```
void fillDps ( str, dps[] )
{  For ( i=0; i<n; i++ )        // n = str.size()
   {  dps[i] = find ( str, i+1 );
   }
}
```

Total T.C = O(n^3)

dps ⇒ O(n^3)

dps[i] =

a b ab ab

dps[3]

i=3

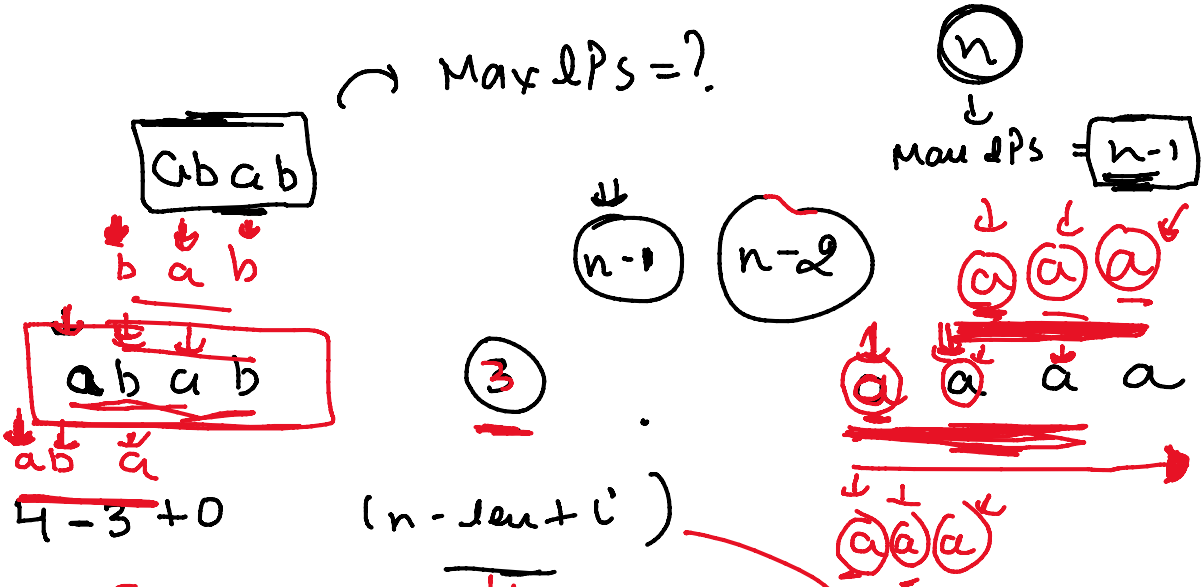
```
int find ( string str, int n )        // n = size of string
{  For ( int len = n-1; len > 0; len-- )
   {  bool flag = true;
      For ( i=0; i < len; i++ )
```

T.C = ?

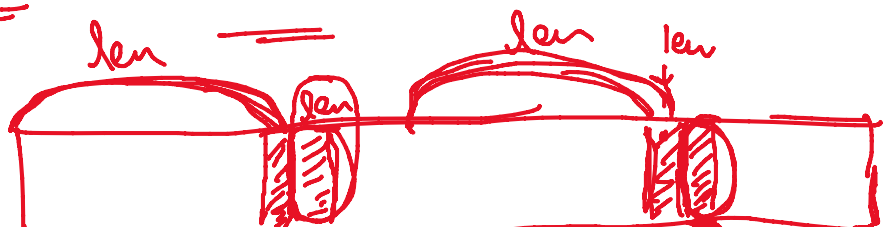
```
for (i=0; i<len; i++)
{   if (str[i] != str[n-len+i])
        flag = false;
}
if (flag)
    return len;
```

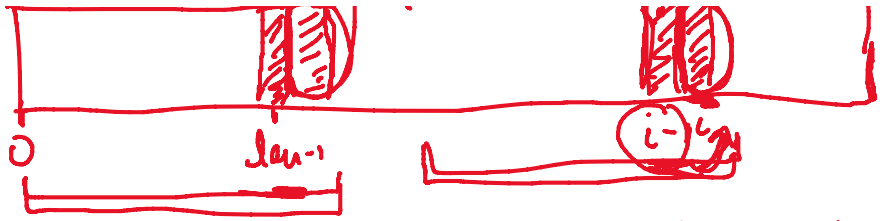
Max length

Max LPS = ?



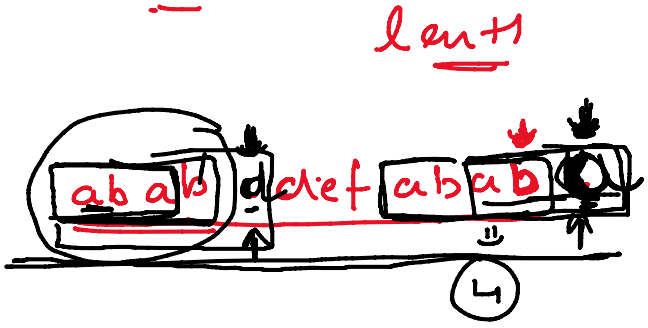
lps[i] = ? starting and ending characters





str[len] == str[i]

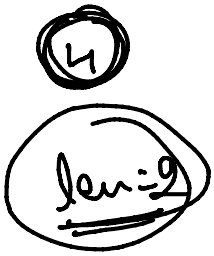
lps[i-1] = len

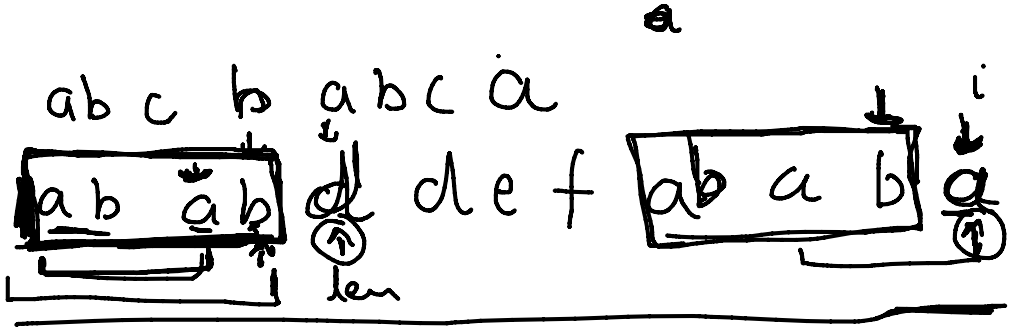


lps[i-1] = lps[i]

lps[i] = 0

ab ab c d





lps[i] = 0?

lps[i] = 3

lps[i] = 3

if (str[len] == str[i]) {

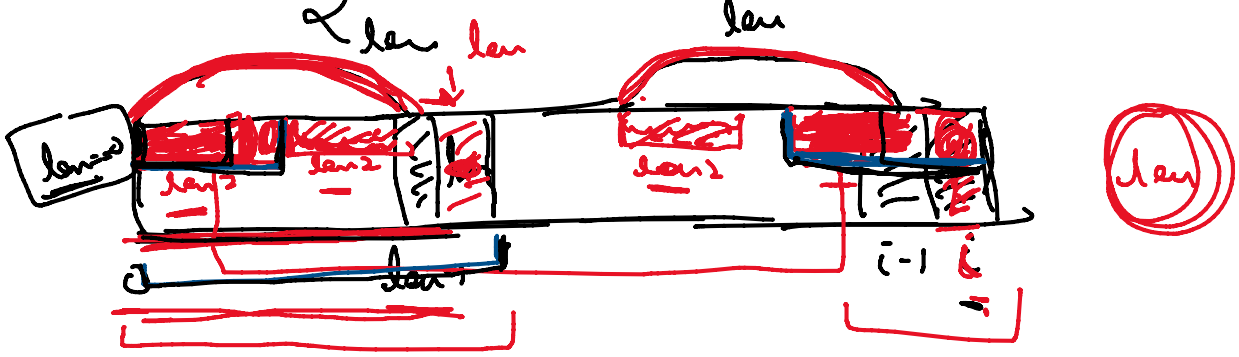
lps[i] = len+1;

len++;

}

len → lps[len-1]

else {



len=2 len+1 ✓

5 3

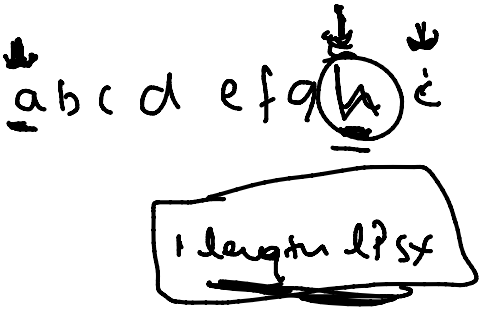
len = lps[len-1]

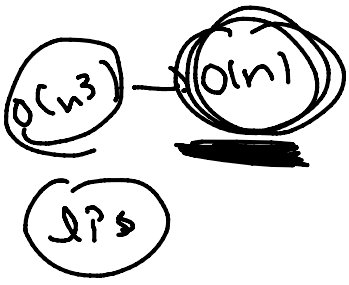
3-4

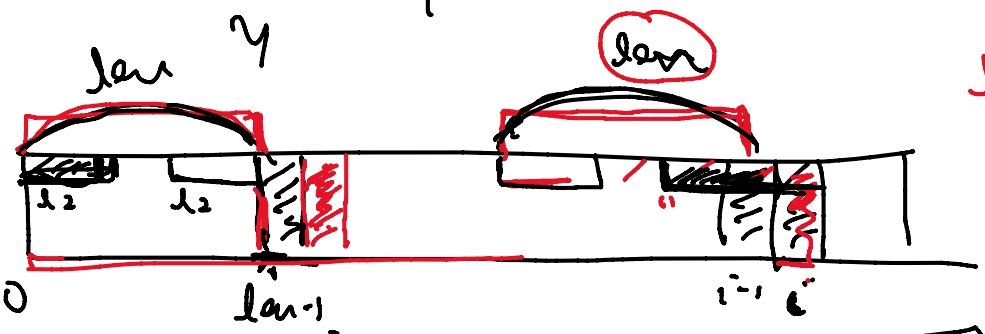
```
void fillLPS ( Str, lps[])
{   int n = str.size();
    int len = 0;
    lps[0] = 0
    i = 1;
    while (i < n)
    {  if ( str[i] == str[len])
       {  len++;
          lps[i] = len;
          i++;
       }
       else
       {  if (len == 0)
          {  lps[i] = 0; i++;
          }
          else
          {  len = lps[len-1];
          }
```

int char lps

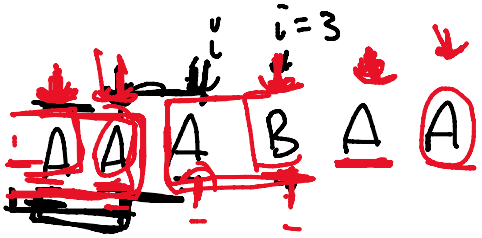
T.C = O(n)







lps[i] = len + 1



n^{2x}

O(n) amortized $n \log x$

lps[7] = [0, 1, 2, 0, 1, 2]

len = 2 → 1

lps = 2? str[3] str[2]

len = lps[len-1]

lps[1]

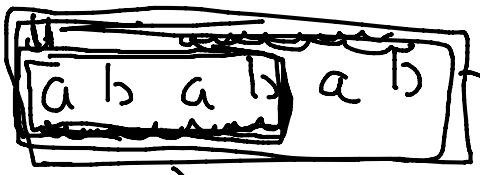
len = lps[1-1] = 0

KMP ⇒ LPS

2-3

LPS?

level ⇒ lps ⇒ l



→ lPs abab

s.substr(0, lps[n-1]);

Same LPS =)

substr (idx, length)

a b c d e f g (1, 3)

bcd

lps[n-1] = 4

String