info

Q) Given a string, find repeated string. → Repeating → ab → Min repeating String

Str: a b c a b c a b → abc

Str2: a b b a b → Ans = abb : string should be a repeatin ab
abababab..*
Ans = abab

Str3: b c d a b c d a b → bc : bcbcbc..*
bcd : bcd bcd bcd – x
y = bcda : b c d a b c d a b *

Str4: abac abac abac
abac abac abac

Str5: aaaaa : 1 Str6: yyyyx → ans : y → yyyy *
yyyy → yyyy *
: y

For a string of len = N of period = x
[0, r-1] : [r, 2r-1] = [2r, 3r-1], ...
r<N
TC = O(N)
$\forall_{i=0}^{N-1}$ s[i%x] == s[i] we don't know x value
Best: 1 Worst: N

$\forall_{r=1}^{N} \forall_{i=0}^{N-1}$ [s[i%x] == s[i]]
TC: $O(N^2)$ SC: $O(1)$

r=1 x
r=2 x
r=3 ✓
:
r=N

Str4: a b a c a b a c a b a c
0 1 2 3 4 5 6 7 8 9 10 11
str4 period

2nd Approach: Assume BS:

Str: a b c a : expected ans = 3
lo hi mid
1 4 2 : a b c
2 4 3

How BS?
we cannot differentiate a part of search space.

Ex: a b c a b a b
ans = abab
abcabcabc *
ans = abcab
abcab ab

Ex2: a b c a b c a b c a
lo hi mid
1 10 5 → 4, 6, 37
mid = 12, 11, 37

Ex: a b c a a b c a a b c a
str[] 0 0 0 0 0 1 0 0 0 1
a b c a a b c a
a b c a a b c a

[0-4] = [4-11]
[0-3] = [4-7] = [8-11] → period: 4

Ex: a b c d a a b c d a a b c d a a b c
str[] 0 0 0 0 1 15 0 0 0 1 0 0 0 1 3 0 0
a b c d a a b c d a a b c d a a b c

[0-12] = [5-17]
0-4 = 5-9 = 10-14 = 15-19 → period = 5
0-4 = 5-9
5-9 = 10-14
10-12 = 15-17

Set
→ Step1: Contains 2 []
→ Step2: find i such that str[i]==x → pattern ?

S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12
0 1 2 3 4 5 6 7
4 5 6 7 8 9 10 11 12

S0 S1 S2 S3 S4 S5 S6
3
3+4=7

Q) Length of longest prefix also suffix for a string.

2nd) String : a b c x y a b c

Prefix: Strings starting at index 0
a
ab
abc
abcx
abcxy
abcxya
abcxyab
abcxyabc

Suffix: String ending at index N-1
TC: (N-1)*O(N)
→ $O(N^2)$
c
bc
abc : 2
yabc
xyabc
cxyabc
bcxyabc
abcxyabc

Ex: String
b ba bab baba babab
0 0 1 2 3

Str : b a b a b
LPS[] 0 0 1 2 3
LPS[i] = Longest prefix string also suffix string Ending at i

→ Given a string N
for N LPS value
→ TC: O(N) * O(N²)
→ O(N²)

observations:

1) LPS[i+1] ≤ 1 + LPS[i]

S0 S1 S2 S3 S4 ... Si-5 Si-4 Si-3 Si-2 Si-1 Si Si+1
≥ 4 5
atleast 4

S0 - S4 = Si-3 - Si+1
S0 S1 S2 S3 S4 = Si-3 Si-2 Si-1 Si Si+1

LPS[i+1] ≤ x
LPS[i] ≥ x-1
LPS[i] ≥ LPS[i+1]-1
LPS[i+1] ≥ LPS[i+1]
LPS[i+1] ≤ LPS[i]+1

if Si Si+1
x x+1
→ Max possible value
→ Max is more possible value

Case1:
0 1 2 3 4 5 6
a b a y a b a y
LPS[] 0 0 1 0 1 2 3 4
0-2 = 4-6

if str[i] == str[i-1]
LPS[i] = LPS[i-1]+1
str[LPS[i-1]] == str[i]
LPS[i] = LPS[i-1]+1

Ingenral
for x = LPS[i-1]
if(str[x] == str[i])
LPS[i] = x+1

Case2:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
c a c a c a c a b c a c a c a c a
LPS[] 0 0 1 0 1 2 3 2 0 1 2 3 4 5 6 7 4
0-6 = 7-15
x = LPS[15] = 7
str[x] == str[16] *
7 = 16
x = 7
x = LPS[7-1]
= LPS[6]
x = 3
3 - 16
str[3] == str[16] ✓ → 2nd comp
LPS[16] = x+1 = 4

Case3:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
a b c a b d a b c a b e a b c a b d a b c a b e
0 0 0 1 2 0 1 2 3 4 5 0 1 2 3 4 5 6 7 8 9 10 11 3

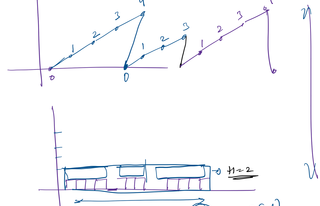
x = LPS[7-1] = 11
str[x] == str[23] str[11] == str[23] *
x = 11
x = LPS[x-1]
= LPS[10] = 5
str[x] == str[23] *
x = LPS[x-1]
= LPS[4]
= 2
str[x] == str[i] = str[x] == str[23] ✓
LPS[i] = x+1

LPS[0] = 0
for(int i = 1; i < N; i++)
// we want to fill LPS[i]
int x = LPS[i-1] → 1
while (x > 0 && str[x] != str[i]) {
x = LPS[x-1]
}
if (str[x] == str[i]) {
LPS[i] = x+1
}
else {
if (str[0] == str[i]) {
LPS[i] = 1
}
}
}

if x==0, we are not comparing.
if (str[x] == str[i]) {
LPS[i] = x+1
}

i = 1; i < N; i++
x = LPS[i-1];
while(x > 0 && str[x] != str[i]){
x = LPS[x-1]
}
if(str[x] == str[i]){
LPS[i] = x+1
}

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
4 3 4
2N



→ Given a string = N, Calculate no. of characters we need to add at start to make the entire string palindrome.

Str: e c a b a c e
5 5-3=2
→ N = length of longest palindrome starting at 0th index
added characters
7

Str2: d c a a b a a c d
→ added
7-4=3

S0 : pal or not
S0S1 : pal or not
S0S1S2 : pal or not
TC: $O(N^2)$

Ex: a b c b a d e f f e d a b c b a
5
→ Goal = str2+str, find LPS of this.
→ differentiate

Reason : Ex: Str = aaa
Ex: aaa aaa
5 → 5